LE CONOSCENZE INDISPENSABILI PER DIVENTARE AMMINISTRATORE DI RETE IN TECNOLOGIA WINDOWS

LAVORARE CON ACTIVE DIRECTORY

Francesco Lippo

EDIZIONI MASTER

i libri di

LAVORARE CON ACTIVE DIRECTORY

di Francesco Lippo

INDICE

Introduzione ad Active Directory

Domini di Windows 2000 e AD

Global catalog, FSMO

LDAP: comunicare con AD

Manipolare gli oggetti di AD

Esempi pratici

INTRODUZIONE AD ACTIVE DIRECTORY

Active Directory, come molti sapranno, è l'implementazione Microsoft di un servizio di directory. Nelle prossime pagine, in particolare, spiegheremo proprio cosa s'intenda per directory e servizi di directory, concentrando la nostra attenzione sugli aspetti essenziali ed utili ad affrontare l'argomento in maniera molto più sicura.

Premessa

Probabilmente il concetto di directory e di servizio di directory è un concetto poco conosciuto o, addirittura per alcuni, del tutto nuovo. Tuttavia, per poter comprendere a fondo cosa sia realmente Active Directory, è necessario avere chiari questi due concetti che risultano essere alla base di tutto il discorso che faremo in seguito.

1.1 IL CONCETTO DI DIRECTORY

Se qualcuno ci ponesse la domanda "cos'è una directory", sicuramente, la prima cosa che ci verrebbe da rispondere, è qualcosa di simile a "un insieme di file e cartelle". Naturalmente questa risposta è alquanto ovvia perché, con il passare degli anni, abbiamo acquisito ed assimilato questo termine molto bene, avendo passato ore ed ore davanti al nostro computer a copiare documenti, aprirli, spostarli, ecc.
In realtà, in questo contesto e per tutto il discorso che faremo in seguito, il termine directory, assume una forma nuova, diversa da quanto appreso in passato e cercheremo di spiegarlo con un piccolo esempio. Immaginiamo di avere una rubrica telefonica. Come tutti sanno, essa non è altro che un insieme di informazioni, contenente i nominativi delle persone e quelli delle aziende, con il loro numero telefonico e, a volte, altre informazioni. Bene: la directory è proprio questo. Per essere più rigorosi, potremmo dire che una directory è un catalogo d'informazioni (come quelle viste in precedenza, ad esem-

pio) elencate e raggruppate secondo un certo criterio, magari tenendo conto della necessità di doverne migliorare l'accesso. Un altro esempio che consente di spiegare questo concetto potrebbe essere quello delle pagine gialle.
A questo punto, se abbiamo compreso il concetto appena esposto, non dovrebbe essere difficile comprendere quello legato al termine servizio di directory.

1.2 SERVIZIO DI DIRECTORY

Un servizio di directory altro non è che un servizio in grado di consentire ad un utente (client) di accedere e/o memorizzare informazioni all'interno di una directory, nel nostro caso computerizzata. Esistono vari tipi di servizi di directory e sono basati o definiscono un protocollo (o una famiglia di protocolli) in grado d'interagire con la directory stessa.
Può sembrare ovvio, ma occorre precisare che i servizi di directory non sono tutti uguali tra loro. La motivazione è abbastanza semplice. Possono esistere servizi di directory realizzati semplicemente per fornire informazioni, come ad esempio, nel caso della semplice rubrica telefonica, il numero telefonico di un utente o il suo codice fiscale. Oppure possono esistere servizi di directory più complessi che si occupano di centralizzare le informazioni sugli utenti, sulle risorse, ecc. fornendo nel contempo stesso, una soluzione scalabile e flessibile, in grado di "assorbire" i compiti di altri servizi di directory e nascondendo all'utente (client) la complessità delle interfacce implementate.
Probabilmente qualcuno si starà chiedendo quale sia la "forma" reale di un servizio di directory e se, volontariamente o no, abbia mai avuto necessità di usarne uno. La risposta alla seconda domanda è "sicuramente sì". Basti pensare al Domain Name System (DNS) o al servizio WINS. Se ci pensate, entrambi questi sistemi rispondono alla definizione più o meno indicativa data in precedenza ad

un servizio di directory. Entrambi, infatti, dispongono di un insieme di informazioni, organizzate in maniera diversa, ma in grado di consentire, attraverso l'apposito servizio di directory, la memorizzazione ed il recupero delle informazioni richieste dal client.

1.3 SERVIZIO DI DIRECTORY VS DB RELAZIONALE

Se abbiamo finalmente un'idea più chiara di cosa si nasconda dietro i due concetti esposti nei paragrafi precedenti, probabilmente ci staremo chiedendo se ha senso o meno parlare di database relazionali in luogo dei tanto "decantati" directory e servizi di directory. A questo punto, è piuttosto importante capire che esiste una differenza sostanziale tra un servizio di directory ed un database relazionale, anche se a prima vista, i due oggetti possono sembrare analoghi e spesso si tende a confonderli tra loro.
Se pensiamo a come è fatto un DB relazionale e a quanto abbiamo "intuito" tramite gli esempi e le definizioni date in precedenza, la prima differenza che appare quasi ovvia, è il fatto che un servizio di directory, rispetto ad un DB relazionale, può essere inteso come una sorta di database specializzato. Con quest'ultimo termine si vuol semplicemente sottolineare che, differentemente da un DB relazionale, le informazioni che possono essere memorizzate nel primo caso possono essere diverse da quelle che invece sono "adatte" al secondo caso. In parole povere, potremmo anche dire che un servizio di directory è un oggetto che "assomiglia" ad un DB relazionale, ma costruito e pensato più per essere letto che per essere scritto.
I servizi di directory, ovviamente, possono essere semplici oppure, se rapportati al numero ed al tipo di richieste che possono soddisfare, anche complessi. Un tipo di servizio di directory semplice è un servizio che ha compiti "banali" da svolgere. Ad esempio, quello del DNS, è uno di questi e ci serve per comprendere meglio la differenza. Il DNS non fa altro che rispondere alle query dei vari client traducendo

nomi host in IP. Siamo d'accordo che probabilmente questo esempio è "semplificato", ma in realtà è proprio quello che succede.
In un servizio di directory complesso, invece, il numero e, soprattutto, il tipo di richieste d'informazioni da gestire, è più complicato. Il servizio di directory deve essere in grado di rispondere su richieste che ritornano dati di vario tipo e, soprattutto, multiple.
Schematizzando, una situazione che faccia vedere la differenza tra questi due tipi, è la seguente:

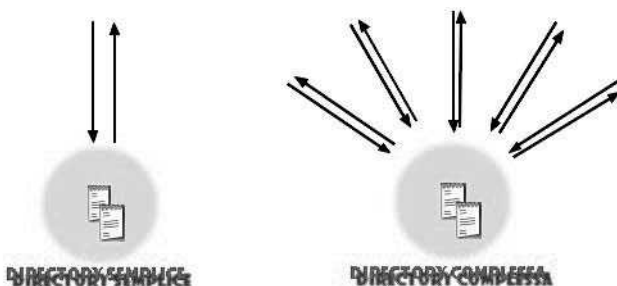



Figura 1: Directory semplice e directory complessa

1.4 WINDOWS 2000 ACTIVE DIRECTORY

Come si sarà finalmente compreso dopo questa breve premessa, Active Directory costituisce il sevizio di directory di Windows 2000.
Considerata l'importanza che un componente di questo tipo ha assunto all'interno di un sistema operativo evoluto come Windows 2000 Server, la Microsoft ha praticamente "centralizzato" tutto attorno a questo importante componente. Infatti, come qualcuno saprà già, malgrado si potesse parlare di servizi di directory anche in Windows NT, con Windows 2000 l'importanza di questo oggetto è notevolmente cambiata a vantaggio dell'intera infrastruttura di rete.
I servizi offerti da Active Directory possono essere considerati come

il centro stella di una rete complessa di informazioni, il fulcro intorno al quale ruotano tutte le informazioni della rete, la fonte unica dei dati. Spesso ci si rende conto che non è facile convincere i neofiti riguardo l'indiscussa importanza di questo componente del sistema operativo, perché chi proviene dall'esperienza di reti NT, può trovarsi disorientato di fronte all'argomento e preferire rimanere all'oscuro su questa nuova tecnologia anziché affrontarla e non facendosi intimorire.A chiunque avesse ancora "paura" di affrontare Active Directory, è importante ricordare che altro non è un database specializzato, un archivio di dati di vario genere.

Conclusioni

Abbiamo finalmente terminato questa infarinatura sui servizi di directory. Abbiamo compreso che Active Directory è il servizio di directory implementato in Windows 2000 ossia un servizio in grado di consentirci l'accesso e quant'altro ad un grande archivio di informazioni.Nel prossimo capitolo affronteremo i concetti principali che ruotano attorno a questo servizio di directory, cercando di capire in maniera più approfondita come funziona e facendo luce su alcuni dei termini tecnici più specifici dell'argomento che ci saranno utili nei capitoli a seguire.

DOMINI DI WINDOWS 2000 E AD

Active Directory, come visto in precedenza, costituisce il servizio di directory di Windows 2000. La sua complessità e, al tempo stesso, l'alta flessibilità con la quale può adattarsi a realtà anche complesse, è legata a diversi fattori e componenti, la cui conoscenza, è di fondamentale importanza per capire la struttura funzionale di questo importante servizio. In questo capitolo spiegheremo concetti importanti come domini, alberi e foreste.

Premessa

Attorno a Windows 2000 ruotano tantissimi concetti importanti. Chi ha esperienza con l'amministrazione di reti Windows noterà una certa familiarità con la terminologia che utilizzeremo in questo capitolo, ma è necessario avere ben chiari almeno i concetti base per poter comprendere i capitoli successivi. Non è certo un'impresa semplice decidere da dove cominciare questo capitolo e sintetizzare quanto più possibile il discorso, ma certamente l'oggetto che meglio si presta ad iniziare questa disquisizione è quello del dominio.

2.1 DOMINI E WORKGROUP

Il concetto di dominio è, senza alcun'ombra di dubbio, il termine che maggiormente sentiamo nominare quando parliamo di reti Microsoft. Per poter comprendere bene cosa intendiamo con questo concetto, facciamo un brevissimo passo indietro e precisamente sino alla comparsa di Windows NT.Sino ad allora, l'installazione di una rete Windows era un'operazione piuttosto "banale". Tutte le macchine erano "alla pari", nel senso che non esistevano computer che svolgevano operazioni particolari, che offrivano servizi ad altre. Questo modello di rete era ed è definito Workgroup e lo possiamo replicare se, ad esempio, installiamo una piccola rete con sole macchine Windows XP.Un dominio, invece, è un concetto del tutto diverso. Possiamo definirlo come un'entità astratta, un insieme di macchine all'interno delle quali esistono computer specializzati a svolgere

determinati compiti e macchine che possono usufruire di questi servizi. Ovviamente, neanche a dirlo, è stato studiato per superare i limiti architetturali offerti dal modello Workgroup.
Dal concetto di dominio arriviamo molto presto alla definizione dell'architettura che identifica posizione e ruoli di ogni macchina al suo interno. Vengono introdotti concetti nuovi e "vecchi" che permettono di definire l'architettura di una rete basata su Windows 2000 ed Active Directory. Ma andiamo per ordine e vediamo più in dettaglio i concetti essenziali.

2.2 I DOMINI DI WINDOWS 2000

Il dominio, lo abbiamo detto precedentemente, può essere considerato come l'oggetto principale, l'unità atomica sulla quale si basa tutta l'infrastruttura di Active Directory. E non poteva essere diversamente se pensiamo che tale concetto, affermatosi dopo con l'avvento di Windows NT doveva necessariamente eliminare i limiti infrastrutturali imposti da un workgroup. In Windows 2000, come in Windows NT, il concetto di dominio non ha subito grosse modifiche dal punto di vista concettuale. Rappresenta sempre un confine amministrativo, in cui però esso stesso rappresenta un namespace, corrispondente, a sua volta, ad un dominio DNS. Il primo dominio che viene creato ha molta importanza e viene chiamato dominio principale perché, da esso, ha origine la struttura gerarchica dalla quale verranno derivati gli altri. Queste strutture sono meglio note come strutture di domini e altro non sono che gruppi di domini di Windows 2000 che, assieme, formano uno spazio nome contiguo. Semplificando, per spazio dei nomi, intendiamo un insieme all'interno del quale un nome può essere risolto. In Active Directory, per default, il dominio rappresenta lo spazio dei nomi per eccellenza ed il processo attraverso il quale un nome viene risolto, è definito come name resolution. Ogni dominio che si trova sotto un altro, viene anche chiamato dominio child. Di seguito un esempio grafico di come può presentarsi una struttura di domini:

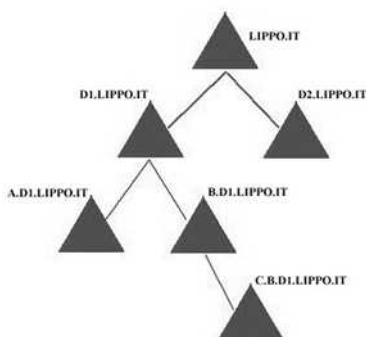



Figura 1: Struttura a domini di LIPPO.IT

La combinazione di una o più strutture di domini costituisce l'insieme di strutture.
Queste strutture riprestano meglio a società complesse ed articolate ed hanno una struttura simile alla figura successiva.
L'importanza di un complesso di strutture così articolato garantisce una certa continuità a società che, pur non avendo/costituendo uno spazio nome contiguo, sono legate tra loro.

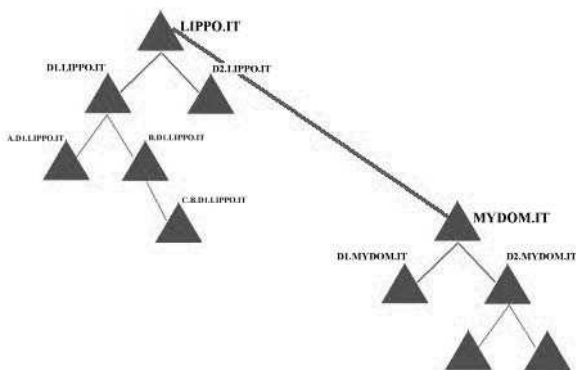



Figura 2: Un esempio d'insieme di strutture

Tutte queste architetture non sono "sufficienti" a definire quanto basta per pianificare una rete basata su Windows 2000.
Infatti, oltre all'ormai noto concetto di dominio, la Microsoft ha introdotto altri due nuovi termini: alberi e foreste.
Un Albero (Tree) è costituito da un insieme di domini che si accordano vicendevolmente la fiducia e che appartengono ad uno spazio di nomi contiguo.
Un esempio che ci può far capire la struttura di un albero è costituita da una struttura composta da dominio e sottodomini del precedente.
Rifacendoci alla notazione precedente, Lippo.it, D1.Lippo.it e A.D1.Lippo.it costituiscono un esempio di albero. Il concetto di Foresta, invece, è leggermente diverso.
Una foresta è costituita da uno o più insiemi di alberi di domini che non condividono uno spazio dei nomi contiguo.
Questo concetto è importante quando si accennerà alla definizione di Schema, poiché tutti i domini di una foresta condividono il medesimo schema. Il dominio di Windows 2000, dunque rimane ancora uno dei limiti amministrativi/organizzativi principali, intendendo con questo che i diritti amministrativi sono "confinati", se non disposto appositamente, all'interno del dominio considerato.
A questo proposito, però, Windows 2000 introduce un nuovo concetto che consente di migliorare la gestione di un intero dominio, consentendo ad un amministratore di delegare alcuni compiti ad altri, facendo sì che il loro "raggio d'azione" all'interno del dominio sia in qualche modo circoscritto ad un suo sottoinsieme.
A questo proposito Windows 2000 introduce il concetto di Organizational Unit (OU).

2.3 LE ORGANIZATIONAL UNIT

Una delle novità di Windows 2000 sono proprio le unità organizzative. Infatti, esse consentono di delegare i controllo delle risorse ad

altri, mantenendo circoscritti i poteri di ciascun amministratore delegato alla propria OU.
Una OU può contenere altre OU i maniera tale da affinare questa pianificazione oppure un insieme di oggetti leaf (oggetti che non possono contenerne altri) come utenti, computer, ecc.
La creazione di una Organizational Unit è possibile, come la maggior parte delle operazioni di questo tipo, utilizzando la MMC e caricando l'apposito snap-in Active Directory Users and Computers.
Per poter delegare ad altri i poteri su di una certa OU, viene utilizzato il Delegation Of Control wizard.
Ecco alcune fasi in figura 3 e figura 4

Figura 3: Delegation of Control wizard

In questa maniera, seguendo le indicazioni del wizard, possiamo liberare l'amministratore da determinati carichi delegando, per l'appunto, alcuni compiti ad altri.

2.4 LO SCHEMA

Dopo aver visto domini, OU, parlato di utenti, gruppi, alberi e foreste, abbiamo sicuramente compreso che Active Directory, in fondo, non è altro che un semplice contenitore di oggetti.

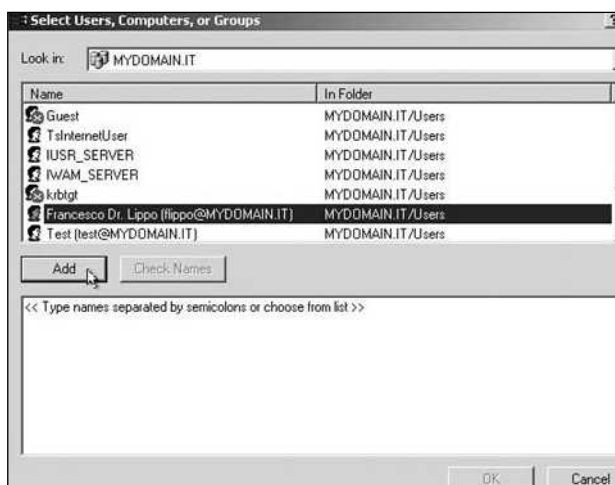



Figura 4: Assegnazione delle delega all'utente

Sebbene sia chiaro a tutti il concetto di oggetto, è bene assegnare a questo termine una definizione più rigorosa che ci consentirà di capire meglio un nuovo componente di AD.
In questo contesto soprattutto,un oggetto può essere definito come un'entità, un elemento in possesso di determinati attributi.
Un utente, un dominio, un gruppo di Windows 2000 sono tutti esempi di oggetti perché, intuitivamente, sappiamo che possiedono degli attributi bene definiti (ad esempio, per gli utenti, un attributo potrebbe essere il nome oppure la password di accesso).
Tuttavia, per ora, la cosa importante che dovremmo chiederci è: chi tiene traccia di questi atrributi, di tutti gli attributi di tutti gli oggetti di Active Directory?
La risposta è lo Schema di Active Directory.
Lo Schema è definito all'interno di AD stesso e può essere visto come tutti gli altri contenitori che conosciamo, anche se è strutturato in maniera tale da definire e descrivere tutte le classi di oggetti contenuti nella directory. Si noti anche che tali informazioni

sono memorizzate nello Schema come se fossero essi stessi degli oggetti.
Per poter dare un'occhiata allo Schema di Active Directory, possiamo utilizzare lo snap-in Active Directory Schema disponibile solo dopo aver installato l'Administration Pack di Windows 2000 e scaricabile dal sito della Microsoft.
In Figura 5 e Figura 6 ecco alcune schermate che danno un'idea di quello che potremmo trovare.

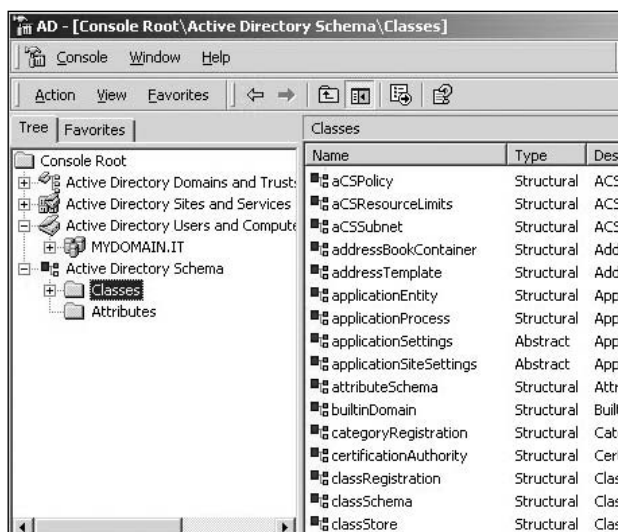



Figura 5: Active Directory Schema Snap-in (1)

Probabilmente è inutile sottolineare l'importanza di questo contenitore ed il fatto che occorre fare molta attenzione ad eventuali modifiche si desideri apportare ai suoi oggetti (operazione possibile solo su di un Domain Controller particolare che ricopre il ruolo di Schema Master, discusso nel successivo capitolo).
Prima di passare oltre, una piccola curiosità: qualcuno si sarà chiesto quale sia il file o i file che rappresentano fisicamente il data-

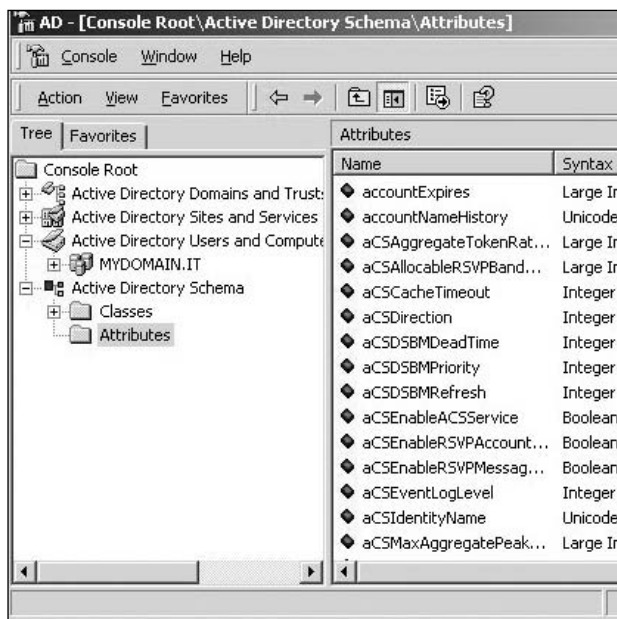



Figura 6: Active Directory Schema Snap-in (2)

base dell'Active Directory. Bene, il file in questione, quello che definisce il DB vero e proprio, è solo uno, ha l'estensione DIT e generalmente è memorizzato in %system%NTDS/ con il nome di ntds.dit.

A titolo puramente informativo, ecco un elenco dei file, in aggiunta al precedente, che nel complesso rappresentano l'intera directory di Windows 2000 sono:

- Edb.log: transaction log;
- Edbxxxxx.log. transaction log ausiliari, utilizzati qualora con il precedente si riscontrassero problemi;
- Edb.chk. checkpoint file per memorizzare lo stato di avanzamento nel processo di spostamento dei dati dal transaction log

al DB di Active Directory;

- Res1.log e Res2.log: reserve log file utili per preallocare spazio disco da utilizzare nel caso si esaurisse inavvertitamente, impedendo la scrittura del DB;
- Temp.edb. in-progress transaction file.
- Schema.ini: utilizzato durante l'operazione d'inizializzazione del DB di Active Directory solo la prima volta nella quale un server è promosso a Domain Controller.

GLOBAL CATALOG, FSMO, …

Quando pianifichiamo una rete Windows 2000, dobbiamo pensare ad essa in funzione della nuova struttura a domini che intendiamo realizzare e considerando che essa è relativamente diversa, in termini architetturali soprattutto rispetto al modello a domini di Windows NT. Tuttavia occorre anche tener presente che esistono operazioni che meglio si prestano al tipo di concezione Windows NT-like e che per questo necessitano di qualche accorgimento in più.

3.1 LE PARTIZIONI DI ACTIVE DIRECTORY

Active Directory rappresenta un servizio di directory piuttosto complesso e, affinché possa rispondere ai requisiti di flessibilità e scalabilità richiesti, si appoggia su di un database che è suddiviso in partizioni.
Ogni Domain Controller contiene sempre due partizioni di base ed uguali per tutti i Domain Controller della Foresta:

- Configuration Partition: conosciuta anche con il nome di Configuration Container contiene informazioni importanti che descrivono la topologia dei siti, delle repliche, ecc.
- Schema Partition: questa partizione rappresenta lo schema del DB, dove sono dichiarate le tutte le classi di oggetti ed i loro attributi. Lo schema, com'è facile intuire, ci da le informazioni utili a capire come sono fatti gli oggetti contenuti nel DB e, quindi, tra l'altro, quali attributi hanno;o).

Ogni DC possiede anche una terza partizione (o replica) che contiene il Domain Naming del dominio al quale appartiene.
Alle tre partizioni appena viste ci si riferisce anche con il termine Naming Context (contesto di denominazione).

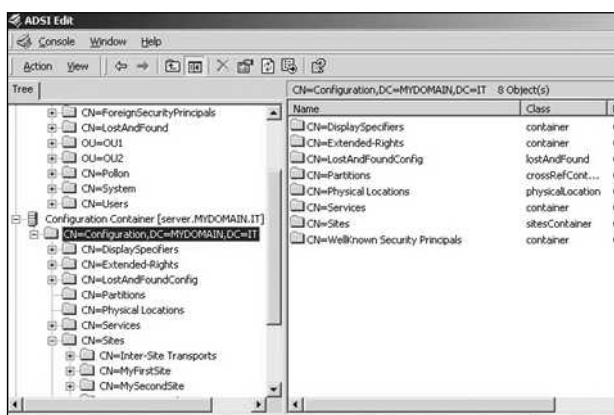



Figura 1: Configuration Container

ADSI Edit
Console Window Help
Action View
Tree
CN=Schema,CN=
Configuration Container [server.MYDOMAIN.IT]
CN=Configuration,DC=MYDOMAIN,DC=IT
CN=DisplaySpecifiers
CN=Extended-Rights
CN=LostAndFoundConfig
CN=Partitions
CN=Physical Locations
CN=Services
CN=Sites
CN=Inter-Site Transports
CN=MyFirstSite
CN=MySecondSite
CN=SiteSenzaSubnet
CN=Subnets
CN=WellKnown Security Principals
Schema [server.MYDOMAIN.IT]
CN=Schema,CN=Configuration,DC=MYDOMAIN,DC=
Domain NC [server.MYDOMAIN.IT]
DC=MYDOMAIN,DC=IT
Name
CN=Account-E
CN=Account-N
CN=ACS-Aggre
CN=ACS-Alloca
CN=ACS-Cache
CN=ACS-Direct
CN=ACS-DSBM
CN=ACS-DSBM
CN=ACS-DSBM
CN=ACS-Enabl
CN=ACS-Enabl
CN=ACS-Enabl
CN=ACS-Event
CN=ACS-Ident
CN=ACS-Max-
CN=ACS-Max-
CN=ACS-Maxin

Figura 2: Schema

3.2 IL GLOBAL CATALOG

Per migliorare le ricerche all'interno di Active Directory, si è pensato di delegare a particolari Domain Controller, il compito di conservare una copia delle informazioni di AD "limitata". Questi server prendono il nome di Global Catalog.
In questo contesto, un catalogo è un indice separato di oggetti contenuti all'interno di una foresta Active Directory che, per default, memorizza solo una parte dei loro attributi. Può non essere evidente, ma un Global Catalog non ha soltanto informazioni relative al proprio dominio, ma possiede anche una replica parziale (quindi, solo alcuni attributi) delle Domain Partitions degli altri domini della foresta.
In questo modo, un GC permette agli utenti di individuare rapidamente, senza obbligarli a chiedere queste informazioni ai Domain Controller. L'importanza di un ruolo di questo tipo risiede anche nel fatto che ogni rete deve avere almeno un Global Catalog possa avvenire l'autenticazione sui domini Active Directory (per default, il "primo controller del primo dominio nel primo albero" diventa un GC.

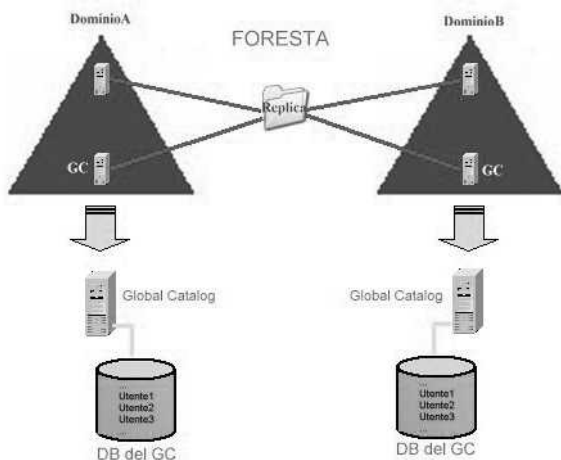



Figura 3: Il database del Global Catalog

3.3 REPLICAZIONE SINGLE MASTER VS MULTIPLE MASTER

Active Directory supporta una tipologia di replicazione delle informazioni contenute nel suo database denominata Multi-Master.
Con questo termine si vuol semplicemente affermare che nessun Domain Controller è "primario" (ossia, usando un termine forse improprio, "proprietario dei dati"), ma, al contrario, una qualunque informazione può essere modificata su qualunque DC per poi essere propagata a tutti gli altri secondo un preciso schema logico di connessioni.
Fermo restando la validità un meccanismo di replicazione siffatto, può non apparire altrettanto evidente che esso possa condurre a conflitti sull'integrità dei dati, qualora una modifica fosse seguita "accidentalmente" e contemporaneamente su più DC allo stesso istante. Ovviamente Windows 2000 ha previsto casi del genere e gestisce tali situazioni mediante l'adozione di un algoritmo che replica solo la modifica effettuata sul DC con data più recente, scartando di conseguenza tutte le altre.
Questo approccio, sebbene possa essere ritenuto accettabile in molti casi, potrebbe non essere adeguato in altre situazioni. Per prevenire conflitti in aggiornamento sugli oggetti vitali all'infrastruttura, Active Directory adotta un modello Single-Master per la modifica di alcune informazioni.
In questa modalità, un solo DC nell'intera directory può essere utilizzato per gli aggiornamenti.
Su questo principio si basa, ad esempio, il Primary Domain Controller di un Dominio Windows NT 4.0, in quanto è l'unico responsabile della modifica dei dati nel proprio dominio. Windows 2000 estende il modello Single-Master, mediante il quale si gestivano gli aggiornamenti in NT, ad altri ruoli e consente di trasferirli su qualsiasi DC dell'infrastruttura.
Siccome i ruoli non sono legati ad un particolare DC, essi sono indicati col termine Flexible Single Master Operation (FSMO) role.

3.4 I RUOLI FSMO DI WINDOWS 2000

Attualmente Windows 2000 prevede cinque ruoli FSMO dei quali 2 riferiti specificamente alle foreste e 3 ai domini:

Ruoli per le foreste

- Master dei nomi di dominio
- Schema Master

Ruoli per i domini

- Emulatore PDC
- Master Infrastructure
- RID Master

Per quanto detto in precedenza, risulta abbastanza ovvio che, per ognuno di essi, può esistere soltanto un server che lo ricopra.
Esistono diversi metodi per stabilire quali siano le macchine che ricoprono ciascuno di questi ruoli ed uno di questi consiste nell'utilizzo dell'utility netdom fornita con il CD di Windows 2000 e localizzata nella cartella \Support\Tools.
Un esempio classico di utilizzo di questo tool è il seguente:

```
netdom query fsmo
```

che restituisce qualcosa del tipo:

```
Schema owner            DCServer.MyDomain.it
Domain role owner       DCServer.MyDomain.it
PDC role                DCServer.MyDomain.it
RID pool manager        DCServer.MyDomain.it
Infrastructure owner    DCServer.MyDomain.it
```

Descriviamoli brevemente:

- **Master dei nomi di dominio:** Il Domain Naming Master è il DC responsabile delle modifiche al Domain Namespace della Directory. Questo DC è l'unico in grado di aggiungere o rimuovere un dominio da Active Directory oltre ad aggiungere o rimuovere cross-reference a domini di altre Directory.

- **Schema Master:** Lo Schema Master rappresenta l'unico DC in grado di apportare modifiche allo Schema di AD. Una volta che lo Schema è stato modificato, esso viene replicato dallo Schema Master agli altri DC nella Directory.

- **Emulatore PDC:** In un dominio Windows 2000 il PDC Emulator ha le seguenti funzioni:

- I cambiamenti di password eseguiti da un DC sono replicati in prima istanza sul PDC Emulator.
- Gestione degli account in stato locked.
- Le mancate autenticazioni ad un dato DC a causa di password errata, sono riverificate sul PDC Emulator prima che venga ritornato un messaggio di login del tipo "accesso non consentito" all'utente.
- Funzionalità di Primary Domain Controller (PDC) per client pre-Windows 2000.
- Funzionalità di time service (necessario al protocollo Kerberos) nel dominio di pertinenza.

In particolare, l'emulatore PDC, in ambienti mixed-mode, agisce da "supporto" ai BDC, mentre in ambienti native-mode si preoccupa di riprocessare le richieste di autenticazione fallite in precedenza sugli altri DC.
Il ruolo appena descritto è probabilmente quello maggiormente coinvolto nelle attività di una rete aziendale soprattutto in ambienti misti, dove sono presenti anche server NT.

Pertanto questo ruolo è quello da tenere maggiormente sotto controllo.

- **RID Master:** Il RID Master è il DC responsabile della gestione delle richieste al RID Pool operate da tutti i DC appartenenti ad dominio. E' inoltre responsabile delle operazioni di "spostamento" di oggetti tra domini differenti.
  Quando un DC crea un security principal object (tipicamente, un nuovo utente o un nuovo gruppo) ad esso viene associato un Security ID (SID) univoco nel dominio. Tale SID costituito da due parti:

- un Domain SID (lo stesso per tutti i SID creati nel dominio);
- un Relative ID (RID) (univoco per ogni security principal SID creato nel dominio).

Ogni DC di un dominio ha a disposizione un pool di RID (costituito da 512 elementi) per consentirgli la creazione di un certo quantitativo di security principal object.
Quando il numero di RID disponibili ad un DC scende sotto una certa soglia, il DC è costretto a generare una richiesta di RID addizionali al RID Master del dominio.
Quest'ultimo risponde a tale richiesta recuperando RID dal pool dei RID di dominio non ancora assegnati e li associa al pool dei RID del DC che ne ha fatto la richiesta.
Tuttavia, è interessante ricordare che il DC non utilizzerà il nuovo insieme di SID ottenuti dal RID Master sino a quando non avrà esaurito quelli che aveva in precedenza a disposizione.

- **Master Infrastructure:** Il Master Infrastructure assicura la consistenza degli oggetti tra il dominio ed i domini remoti. Volendo semplificare, potremmo anche dire che è l'unico server che tiene traccia di oggetti contenuti all'interno

di un'altra directory, ma che risultano essere referenziati anche in quella corrente.

Sappiamo tutti che se un oggetto viene referenziato da un dominio diverso da quello di appartenenza, questa referenza contiene:

- il Globally Unique IDentifier (GUID)
- il Security IDentifier (SID)
- il Distinguished Name (DN) dell'oggetto.

L'ultimo termine, in particolare, ci sarà più chiaro più avanti.
Se l'oggetto referenziato viene spostato, il Master Infrastructure si preoccupa di aggiornare i SID e i DN nelle referenze cross-domain a quell'oggetto.
Per semplicità: se aggiungiamo un utente appartenente ad un do-

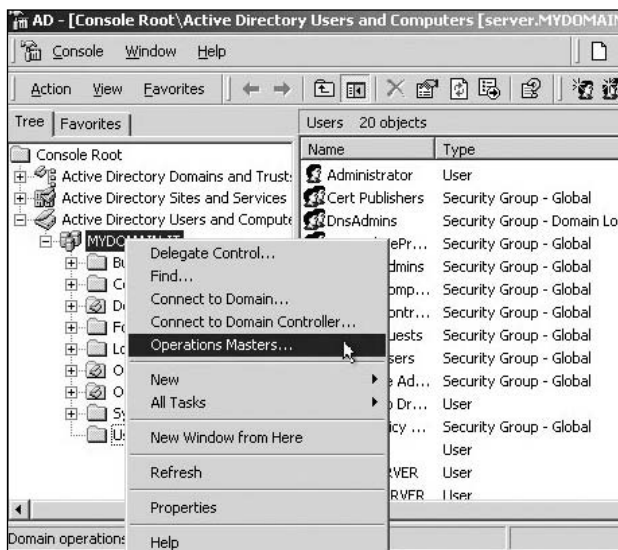



Figura 4: Spostamento degli FSMO Roles da MMC

minio all'interno di un gruppo di sicurezza facente parte di un dominio remoto, il Master Infrastructure si preoccupa di supervisionare l'operazione accertandosi che avvenga nella maniera corretta. Un ruolo siffatto deve essere attivato necessariamente su un DC che non sia il Global Catalog (GC).
La motivazione, sebbene apparentemente "strana", è semplice: se questo ruolo fosse ospitato da un server GC, esso non aggiornerebbe le informazioni sulle relazioni cross-domain a causa del fatto che esso non contiene alcuna referenza ad oggetti che non possiede.Questo perché, lo ricordiamo, un GC possiede solamente una replica parziale degli oggetti della foresta.Tuttavia, nel caso in cui tutti i DC siano allo stesso tempo GC o ci si trovi in presenza di un unico dominio, tale considerazione è superflua. Di seguito un'immagine relativa ad operazioni sugli FSMO roles, utilizzando la MMC.

3.5 ACTIVE DIRECTORY TOOLS

Per poter effettuare operazioni con Active Directory, ma soprattutto per essere certi che le informazioni che recuperiamo tramite i nostri script o che impostiamo manualmente, siano andate a buon fine, possiamo avvalerci di moltissimi strumenti. Ovviamente, neanche a sottolinearlo esplicitamente, la conoscenza dell'esistenza di questi tool, è molto importante perché ci aiuta a comprendere meglio gli oggetti che risiedono all'interno di Active Directory ed a compiere con maggiore sicurezza i nostri test. Malgrado la lista non sia certamente esaustiva, ecco quello che serve per iniziare a fare un pò di pratica:

- Administrative Tools ed in particolare:
- Active Directory Domain and Trusts: consente di intervenire su domini e relazioni di fiducia;
- Active Directory Sites and Services: consente d'intervenire sulla replicazione dei dati;

- Active Directory Users and Computers: permette di manipolare gli oggetti di AD ;
- Active Directory Schema Manager (snap-in disponibile dopo l'installazione dell'AdminPak di Microsoft);
- ADSIEdit: questo tool, utilizzabile sottoforma di snap-in mediante la MMC consente di "sfogliare" le partizioni di Active Directory consentendoci di visualizzare oggetti e attributi in maniera semplice, utilizzando le Active Directory Service Interfaces (ADSI), oggetto di capitoli successivi. Quest'utility è installata come Support Tool. Occorre pertanto servirsi del file Setup.exe contenuto all'interno del CD d'installazione in corrispondenza della cartella Support\Tools.

Tra i tool a linea di comando, disponibili per la maggior parte all'interno del Resource Kit di Windows 2000, ricordiamo innanzitutto Ntdsutil (probabilmente il più importante tool da linea di comando per AD) oltre a:

- AdFind: Active Directory query tool.
- AdMod: Active Directory Modification tool.
- ATSN: IP to Subnet/Site Information tool.
- FindExpAcc: Ricerca account scaduti e con password scaduta.
- GCChk: consente di effettuare verifiche sui Global Catalog.
- MemberOf: mostra i gruppi di appartenenza di un utente.
- SecData: visualizza le security info di utenti e computer.
- Unlock: AD Domain unlock Tool.

Sottolineiamo, ancora una volta, che la lista proposta non è affatto esaustiva e serve esclusivamente per suggerire e mostrare alcune delle utility disponibili.
Prima di passare al capitolo successivo, mostriamo graficamente i passi che occorre eseguire per passare dalla modalità mista a

quella nativa. Quest'ultima modalità non permette di far coesistere domini Windows NT e domini Windows 2000 ed, inoltre, è irreversibile. Tuttavia, introduce (consente) di utilizzare appieno le potenzialità di questo sistema operativo, dando, per esempio, la possibilità di creare i nuovi gruppi universali.

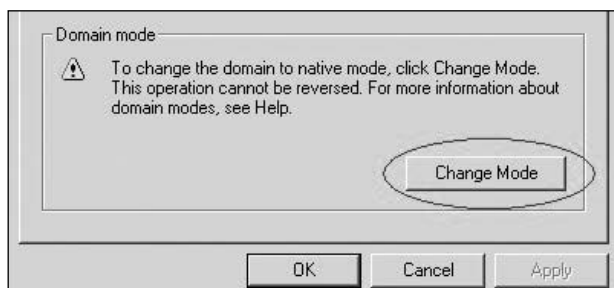



Figura 5: Passaggio alla modalità Native

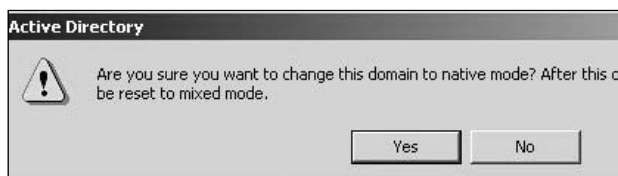



Figura 6: Messaggio di warning sull'irreversibilità dell'operazione

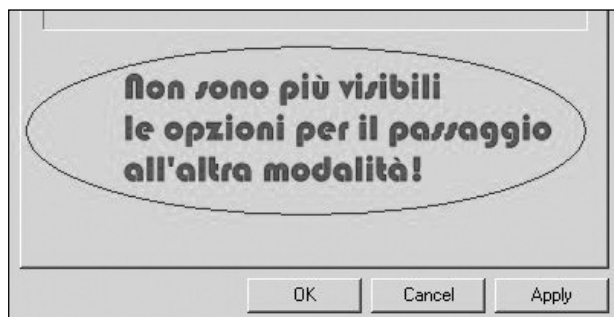



Figura 7: Verifica del passaggio alla modalità Native

LDAP: COMUNICARE CON AD

Come qualunque tipo di database, anche quello su cui si appoggia Windows 2000 Active Directory necessita di un protocollo, un linguaggio che consenta al generico client d'interrogarlo o modificarne il contenuto. Questo protocollo, utilizzato proprio per i servizi di directory e conforme allo standard X.500, esiste ed il suo acronimo è LDAP. Come vedremo, LDAP consente di rappresentare ed individuare qualunque oggetto all'interno di una directory compatibile con esso e la comprensione delle conoscenze di base ci consentirà di capire gli script che mostreremo nei capitoli successivi.

Premessa

Il protocollo LDAP (Lightweight Directory Access Protocol) è uno standard aperto per l'erogazione di servizi di directory tramite una rete Intranet o Internet. Come detto all'inizio, è basato sullo standard X.500 e sul protocollo TCP/IP e rappresenta un'evoluzione del protocollo DAP.

4.1 LA GESTIONE DELLE INFORMAZIONI

Innanzitutto, la prima cosa da dire circa la gestione delle informazioni attraverso LDAP è che esso organizza questo insieme di dati sottoforma di una struttura gerarchica, che logicamente ricorda molto quella utilizzata per rappresentare la struttura di un DNS. L'albero gerarchico che rappresenta queste informazioni è denominato DIT ossia Directory Information Tree.

Questa struttura gerarchica è basata sul concetto di entry. Quest'oggetto è costituito da un insieme di coppie (attributo, valore) e può possedere una sola entry-genitore. Non esistono limiti, invece, al numero di figli di ciascuna entry. Inoltre potremmo anche definirla come una raccolta di attributi che fanno riferimento ad un Distinguished Name (DN).

Il Distinguished Name è uno dei termini che incontremo spesso durante questo cammino e quindi, come tutti quelli che avremo occasione di mostrare, va compreso bene. Accanto ad essi, esistono altri termini importanti e cercheremo di chiarire le idee partendo da un esempio. Immaginiamo di avere di fronte la struttura parziale di un albero DIT relativo ad un dominio Windows 2000 (vedi Figura 33).
Facendo riferimento alla figura successiva, possiamo comprendere meglio le definizioni seguenti:

- Distinguished Name (DN): è tale una stringa (nome) che identifica il percorso da una entry (foglia) dell'albero DIT verso la radice. Un esempio è "LDAP://CN=Francesco Dr. Lippo,CN=Users,DC=MyDomain,DC=it" che individua un utente (Francesco Dr. Lippo) del dominio MyDomain.it;
- Relative Distinguished Name (RDN): qualunque nome di entry, sprovvisto di path o che possegga un path parziale è da considerarsi un RDN ad es. è CN=Francesco Dr. Lippo.

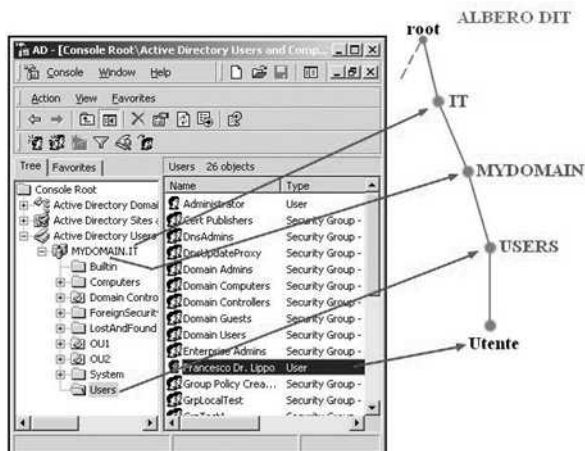



Figura 1: Generico albero DIT

Ora possiamo anche aggiungere che un Distinguished Name può essere considerato come una sequenza ordinata di Relative Distinguished Name, letti dalla generica foglia dell'albero DIT sino alla radice.Soprattutto chi non ha dimestichezza con LDAP, si starà chiedendo che significato abbiano le varie CN e DC contenute all'interno della generica stringa DN. Vediamo di capirne il significato con un piccolo esempio.

4.2 I COMMON NAME

Immaginiamo di visualizzare nella nostra mente il generico albero DIT come un qualsiasi filesystem, laddove possiamo trovare decine di oggetti (file) a partire dalla root e contenitori (directory) che possono contenere non solo altri file, ma addirittura altre cartelle. Se riflettiamo un attimo alla nostra partizione C:, ad esempio, ci appare quasi ovvio che:

- uno stesso file, con lo stesso nome, all'interno di una stessa cartella, non può esistere;
- possiamo avere più file con lo stesso nome presenti su C: purchè siano contenuti in cartelle diverse;
- una cartella può contenere altre cartelle, purchè, allo stesso livello, non esistano directory con lo stesso nome;
- il percorso di un generico file più il nome del file stesso, identifica quel documento in maniera univoca;

Se pensiamo ad un generico albero DIT come ad una struttura simile a quella di un filesystem, le affermazioni appena citate, devono essere mantenute valide ossia:

- ogni oggetto deve avere il nome diverso da quelli presenti nel suo stesso contenitore;
- possono esistere oggetti con lo stesso nome, purché presenti in

contenitori diversi;
- un oggetto-contenitore ne può contenere altri purché abbiano tutti nomi diversi;
- ogni oggetto deve poter essere identificato in maniera univoca.

L'ultimo punto, come visto all'inizio, è soddisfatto dal Distinguished Name che ci assicura che un oggetto identificato dallo stesso DN non può esistere. Per gli altri punti, invece, consideriamo il caso di un generico file presente sul nostro computer.
Comprediamo facilmente che se scriviamo C:\Windows\MioDocumento.doc identifichiamo in maniera univoca il file MioDocumento.doc contenuto nella cartella Windows.
Ci appare quasi ovvia la distinzione, all'interno della stringa di cui sopra, tra la cartella (Windows) ed il file (MioDocumento.doc).
Tuttavia in LDAP abbiamo bisogno di qualche accorgimento in più, soprattutto a causa dei diversi tipi di oggetti e di contenitori con cui avremo a che fare.
Riprendiamo in considerazione l'RDN:

```
CN=Francesco
```

Questo RDN specifica che l'attributo cn del nostro oggetto è "Francesco".
Questo attributo è meglio conosciuto come Common Name e definisce proprio l'attributo che da il nome a quell'oggetto.
Se, quindi, scriviamo CN=Francesco, non abbiamo fatto altro che chiamare per nome un oggetto contenuto all'interno di un qualunque contenitore. Abbiamo detto però che potremmo avere anche due oggetti con lo stesso nome, purché siano su "livelli" diversi dell'albero DIT. Ad esempio, se parliamo di utenti, potremmo avere due utenti Pippo il cui Common Name sia:

```
CN=Pippo
```

ma il cui Distinguished Name, necessariamente diverso, potrebbe essere:

```
CN=Pippo,OU=OU1,DC=MyDomain.it
CN=Pippo,OU=OU2,DC=MyDomain.it
```

A questo punto però abbiamo compreso che per identificare l'utente Pippo, dobbiamo scrivere CN=Pippo e qualcuno potrebbe chiedersi il perché di questa complicazione visto che potevamo "semplicemente" indicarlo nell'RDN come Pippo senza il prefisso CN=.
La spiegazione è molto semplice e per comprenderla ci rifaremo di nuovo al parallelo con i file. I file MioDocumento.xls e MioDocumento.doc possono coesistere all'interno di una stessa cartella, perché il loro nome è "differenziato" dall'estensione.
Per gli oggetti di un albero LDAP, il discorso è analogo.
Per queste ragioni, sono corretti i seguenti RDN all'interno di uno stesso contenitore:

```
CN=Guest
OU=Guest
```

In definitiva possiamo vedere il Common Name un pò come le estensioni dei file che, oltre a distinguere un file da un altro, consentono di definirne il nome.In LDAP, ogni classe di oggetti definisce quale sarà l'attributo che definirà il nome per tutte le istanze di quella classe.
Nel caso di Active Directory abbiamo, in aggiunta a CN:

- OU: Organizational Unit;
- DC: Domain component.

Active Directory, in aggiunta a CN, OU e DC, riconosce e supporta anche O (Organization), ma non la utilizza.

4.3 ACTIVE DIRECTORY SERVICE INTERFACES

Siamo finalmente arrivati all'argomento principe di questo libro ossia Active Directory Service Interfaces (ADSI), un insieme d'interfacce COM disponibili con Windows 2000 ed in grado di consentire ad un qualunque applicativo (linguaggio di programmazione) che supporti questa tecnologia, d'interfacciarsi ad Active Directory (e non solo) interrogarne il DB o manipolarne i dati.

4.4 ADSI

Active Directory Service Interface rappresenta un set d'interfacce COM nata dall'esigenza d'interfacciarsi ad una directory, nascondendo la complessità all'utente stesso.
Come vedremo, l'architettura di ADSI si basa su di un modello particolare denominato Provider-Model ossia un modello attraverso il quale, il generico client deve solo preoccuparsi dell'interfacciamento verso gli oggetti COM esposti da ADSI e lasciando il resto del lavoro ai provider specializzati. Per poter comprendere l'architettura sulla quale si basa ADSI e, in particolare per comprendere in che maniera può tornarci utile per recuperare informazioni da Active Directory, partiremo dal risultato che vogliamo ottenere.
Immaginiamo di voler leggere le informazioni contenute all'interno di un determinato database.
Sicuramente, per raggiungere il nostro scopo, dovremmo compiere alcune fasi, sintetizzate e semplificate appositamente in:

- Collegamento al DB;
- Avvio della query (nel caso di ricerche) specificando la "tabella" di riferimento ed i campi oggetto della ricerca;
- Lettura dei risultati.

Nel caso di manipolazione dei dati, i passi non sono poi così diversi

da quanto appena indicato. I punti importanti sono, in particolare: la conoscenza del path e del nome del DB (utile per potersi collegare ad esso), il nome della tabella, l'elenco dei campi e qualche altra informazione come, ad esempio, le relazioni tra le tabelle.
Active Directory, come già spiegato, è un servizio di directory ossia un oggetto che opera su di un DB specializzato.
Questo significa che qualora desiderassimo leggere o scrivere informazioni all'interno del suo database, non dovremmo compiere operazioni poi molto diverse da quanto visto in precedenza per un generico archivio informatico. Innanzitutto, quindi, occorre trovare un Domain Controller e agganciare l'oggetto desiderato presente all'interno della directory.
Quest'operazione è meglio nota con il termine di binding ed è la prima operazione che deve essere compiuta quando si vuol lavorare con gli oggetti di una directory. ADSI, come preannunciato diverse volte, rappresenta semplicemente una suite di oggetti COM che consentono ad un generico client di lavorare, nel nostro caso specifico, con Active Directory facendo sì che:

- Il client si rivolga a queste interfacce per le sue richieste;
- il provider specifico si preoccupi dell'interfacciamento con la directory, traducendo le richieste del client e nascondendo, quindi, tutto il livello sottostante al client stesso.

Graficamente, la situazione che meglio rappresenta quest'architettura, è mostrata di seguito in (figura 2). Le ultime considerazioni fatte, possono sembrare poco importanti, ma se abbiamo prestato attenzione a quanto sinora detto, si evince facilmente che tutto ciò è di enorme comodità perché ci libera dal dover conoscere i determinati "dettagli" di una certa directory.
A questo punto, manca certamente un pezzo per poter concludere, almeno in parte, questo discorso ossia il provider che si occupa di completare quest'architettura.

ADSI permette di lavorare con diversi tipi di servizi di directory e, quindi, può servirsi di diversi tipi di provider utili allo scopo.

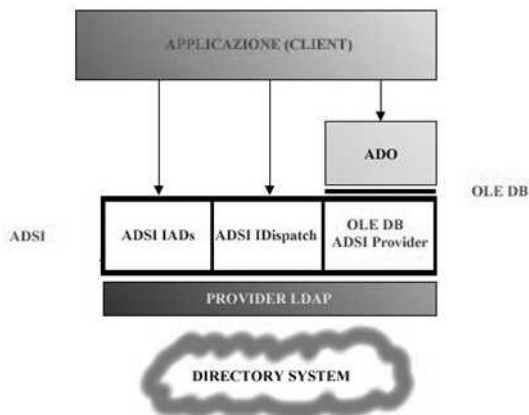



Figura 2: Architettura su cui si basa ADSI

Le ultime considerazioni fatte, possono sembrare poco importanti, ma se abbiamo prestato attenzione a quanto sinora detto, si evince facilmente che tutto ciò è di enorme comodità perché ci libera dal dover conoscere i determinati "dettagli" di una certa directory.
A questo punto, manca certamente un pezzo per poter concludere, almeno in parte, questo discorso ossia il provider che si occupa di completare quest'architettura. ADSI permette di lavorare con diversi tipi di servizi di directory e, quindi, può servirsi di diversi tipi di provider utili allo scopo. In particolare, abbiamo:

- GC: utilizza una porta di comunicazione diversa dall' LDAP provider, ma è molto simile in termini di funzionamento ed utilizzo. Un'ulteriore differenza sta nel fatto che accede al Global Catalog di Active Directory;
- IIS: consente l'accesso all'Internet Information Services (IIS) me-

tabase;
- LDAP: come accennato, consente di lavorare con qualunque tipo di directory compatibile con LDAP (tra cui Active Directory);
- NDS: consente l'accesso ai servizi di directory Novell;
- NWCOMPAT: consente l'accesso ai servizi di directory Novell 3.x;
- WinNT: permette di accedere ad informazioni contenute nel SAM di Windows NT .

In conclusione, il binding ad un oggetto di Active Directory avviene sfruttando l'LDAP Provider. Il Lightweight Directory Access Protocol, come accennato nel capitolo precedente, è un protocollo standard che viene utilizzato proprio per l'accesso a servizi di directory. Esso è conforme allo standard X.500, ma risulta molto più semplice e "leggero" da implementare. Vediamo dunque come utilizzarlo per interrogare Active Directory.

4.5 IL BINDING IN PRATICA

L'operazione di binding compiuta attraverso ADSI verso un oggetto della directory, si conclude attraverso la creazione di un oggetto COM appropriato attraverso il quale, sfruttando metodi e proprietà, potremo agire sugli oggetti di Active Directory.
Ogni oggetto ADSI può implementare diversi tipi d'interfacce, alcune della quali prestabilite ed altre specifiche del tipo di oggetto che stiamo consideranto. Per quanto riguarda le interfacce della prima categoria, sottolineiamo che ogni oggetto ADSI implementa almeno le seguenti interfacce:

- IADs
- IADsContainer
- IADsDeleteOps
- IADsObjectOptions

- IADsOpenDSObject
- IADsPropertyList
- IDirectoryObject
- IDirectorySearch

Inutile ribadire che il numero ed il tipo d'interfacce esposte da ogni oggetto ADSI-COM dipende dal tipo di oggetto che si sta rappresentando e dal provider utilizzato. Più avanti in questo capitolo, mostreremo un elenco maggiormente dettagliato sulle interfacce disponibili, cercando di capire quali siano quelle maggiormente importanti per poter iniziare a lavorare con Active Directory. Ritorniamo dunque al nostro problema, quello del binding e consideriamo i metodi messi a disposizione da COM che ci permettono di portare a termine un'operazione di binding. Non menzioneremo quelli relativi a linguaggi diversi da Visual Basic e VBScript, rimandando per maggiori dettagli in proposito alla documentazione Microsoft:

- *GetObject*: le credenziali utilizzate sono quelle dell'utente corrente (metodo principalmente usato nel libro per motivi di semplicità);
- *IADsOpenDSObject*: è possible specificare credenziali alternative.

A questo punto abbiamo lo strumento per "agganciare" un particolare oggetto di Active Directory, abbiamo individuato (anche se parzialmente) il provider che ci interessa, ma manca ancora qualcosa. Infatti, non abbiamo ancora visto in che maniera indicare al provider qual è l'oggetto su cui poter lavorare. Active Directory possiede migliaia di oggetti ed occorrerebbe un percorso preciso ed univoco da indicare alla GetObject() o funzioni analoghe, per individuare una stringa che ci consenta di raggiungere quello che vogliamo senza possibilità di equivoci. Questa stringa, ovviamente, esiste e prende il nome di ADSPath.

4.6 LA STRINGA ADSPATH

Facendo riferimento al provider LDAP necessario per la maggior parte delle operazioni che vedremo, ecco la generica forma della stringa ADSPath:

```
LDAP://Nome Host/ObjectName
```

dove:

- Nome Host: è il nome del server al quale si sta facendo riferimento. Il suo inserimento è opzionale;
- ObjectName: oggetto specifico al quale ci stiamo riferendo. Per indicarlo possiamo sfruttare diverse notazioni ossia il suo Distinguished Name, il suo Canonical Name oppure, addirittura, il suo GUID.

Quindi, un ADSPath string è composta da almeno 4 componenti ossia il nome del provider (il cosiddetto progID), i caratteri "://", il nome dell'host e l'oggetto al quale ci stiamo riferendo (espresso secondo la sintassi prevista dal provider namespace. Tanto per farci un'idea ancor più precisa, vediamo alcuni esempi, riferiti, in particolare, al provider LDAP:

```
LDAP://CN=Guest,CN=Users,DC=EdMaster,DC=it
LDAP://CN=Francesco Lippo,CN=Users,DC=EdMaster,DC=it
LDAP://OU=OU2,OU=OU1,DC=Lippo,DC=it
```

Andando avanti con questo argomento avremo modo di vedere diverse altre stringhe relative ad ADSPath ed impareremo conoscere meglio l'argomento. Tuttavia, prima di passare oltre, è bene sottolineare ancora un'altra cosa.
Quando specifichiamo la stringa ADSPath in un'operazione di binding ad oggetti di Active Directory, a meno di non avere necessità parti-

colari, è buona norma non specificare mai il server di riferimento per non essere "vincolati", tra le altre cose, nell'operazione.
Infatti, a tal proposito occorre sottolineare che Active Directory supporta il server-less binding che, come si sarà intuito, garantisce il binding ad un oggetto di AD senza necessariamente specificare il server al quale agganciarsi.

4.7 SERVER-LESS BINDING

Qualcuno se ne sarà certamente accorto: all'interno degli esempi di ADSPath string precedenti manca il riferimento al server.
Malgrado la sintassi utilizzata per costruire la stringa di riferimento all'oggetto di AD lo prevedesse, laddove possibile, è preferibile/possibile non indicarlo all'interno dei propri programmi o script per diverse ragioni.
La prima fra tutte è quella di non "vincolare" il codice realizzato a quel particolare server (per maggiore flessibilità); il secondo perché è necessario tener conto dell'eventualità che il server indicato sia irraggiungibile.
Infatti, utilizzando questa modalità, il binding avviene a livello di namespace e non di domain controller.
Questo significa che il locator service si preoccupa di rintracciare un domain controller per effettuare l'operazione di binding, prescindendo dall'uno o dall'altro Domain Controller.

4.8 L'OGGETTO ROOTDSE

Abbiamo ancora alcune considerazioni da fare sul binding, prima di passare a nuovi argomenti. In particolare, va premesso che, qualora si lavori con uno o poco più di un dominio, il server-less binding funziona molto bene, ma le cose, almeno dal punto di vista del programmatore, si complicano se vogliamo realizzare script in grado di essere riutilizzati in ambienti più complessi.

Con questa premessa non si vuol dire che quanto finora sottolineato non va bene, anzi, occorre soltanto aggiungere un nuovo elemento alla nostra "ipotetica" stringa ADSPath che può tornarci utile: RootDSE.L'oggetto RootDSE (l'acronimo DSE, in fondo al nome dell'oggetto, sta per DSA-Specific Entry) ha lo scopo di fornire informazioni sul directory server e consente di eliminare il riferimento ad un particolare dominio, permettendoci di realizzare script e programmi flessibili ed in grado di funzionare su ambienti con più di un dominio.
In LDAP 3.0, RootDSE identifica, se vogliamo, la root dell'albero gerarchico che rappresenta le informazioni delle directory e fornisce informazioni utili sulla sua struttura.
Di seguito alcuni semplici esempi che mostrano il riferimento all'oggetto RootDSE nelle stringhe ADSPath sfruttando le notazioni sinora viste:

```
LDAP://MyDomainController/RootDSE
LDAP://RootDSE
```

Ovviamente, come ogni oggetto, può essere "agganciato" con istruzioni analoghe a:\

```
Set objRootDSE = GetObject("LDAP://RootDSE")
```

L'oggetto RootDSE possiede ovviamente delle proprietà, attraverso le quali possiamo individuare le informazioni che ci occorrono sul dominio dal quale stiamo lanciando l'applicativo ed, in generale, sul directory sever. Vediamole brevemente.

4.9 PROPRIETÀ DELL'OGGETTO ROOTDSE

Di seguito sono elencate alcune delle proprietà di quest'oggetto:

Proprietà	Descrizione
defaultNamingContext	Distinguished Name del dominio al quale il server "corrente" appartiene
dnsHostName	DNS address per il directory server.
namingContexts Schema,	Array di valori contenenti i Distinguished Name di tutti i naming context memorizzati nel directory server. Nel caso diWindows 2000, per default, un domain controller ne contiene almeno 3: Configuration ed uno relativo al dominio al quale il server appartiene.
rootDomainNamingContext	Distinguished name per il primo dominio della foresta che contiene il dominio al quale il directory server appartiene
schemaNamingContext	Distinguished Name per lo schema container.
serverName	Distinguished Name del server.

Tabella 1: Alcune proprietà dell'oggetto RootDSE

Queste sono soltanto alcune delle proprietà esposte da questo importante oggetto, ma negli script che seguiranno, avremo modo di vederne altre e comunque, l'Appendice A elenca tutti gli attributi di questo oggetto. Tanto per avere un'idea di quello che possiamo ottenere, vediamo qualche riga di codice ed il risultato che otteniamo, ipotizzando di lanciare lo script da una macchina di nome Server, all'interno del dominio MyDomain.it. Per il momento non ci soffermeremo sui metodi richiamati, ma semplicemente sulle proprietà di questo oggetto e sui relativi valori.

```
Option Explicit
Dim objDSE
Set objDSE = GetObject("LDAP://rootDSE")
Wscript.Echo objDSE.Get("defaultNamingContext")
```

```
Wscript.Echo objDSE.Get("dnsHostName")
Wscript.Echo objDSE.Get("serverName")
```

che produce qualcosa del tipo:

```
C:\WINNT\System32\cmd.exe

C:\>cscript "C:\Documents and Settings\Administrator\Desktop\Lista Proprie
un utente.vbs"
Microsoft (R) Windows Script Host Version 5.1 for Windows
Copyright (C) Microsoft Corporation 1996-1999. All rights reserved.

DC=MYDOMAIN,DC=IT
server.MYDOMAIN.IT
CN=SERVER,CN=Servers,CN=MyFirstSite,CN=Sites,CN=Configuration,DC=MYDOMAIN,

C:\>
```

Figura 3: Risultato dello script sulle proprietà di rootDSE

Se alle righe precedenti aggiungiamo, quindi, l'istruzione:

```
strADsPathToDomain = "LDAP://" & objDSE.Get("defaultNamingContext")
Set objDomain = GetObject(strADsPathToDomain).
```

non avremo fatto altro che il binding al dominio di appartenenza (quello dal quale l'utente ha fatto logon), mentre con:

```
strADsPathToRootDomain = "LDAP://" &
objDSE.Get("rootDomainNamingContext")
Set objRootDomain = GetObject(strADsPathToRootDomain)
```

non avremo fatto altro che il binding al root domain di una foresta, indipendentemente dal server dal quale l'utente ha effettuato l'accesso. Queste poche righe, dunque, ci dimostrano come si possa facilmente accedere alle proprietà ed agli oggetti di qualsiasi dominio prescindendo dal nome stesso, ma ricavandoselo grazie all'og-

getto rootDSE. ADSI, possiede diverse interfacce COM, classificate in base all'importanza ed agli scopi per cui sono state progettate. Tra le più importanti va menzionata sicuramente la categoria definita Core, alla quale appartiene l'interfaccia IADs e dalla quale derivano proprietà e metodi base di ogni oggetto ADSI.
Ecco un elenco ordinato delle interfacce ADSI disponibili:

Interfaccia	Categoria
IADs	Core
IADsAccessControlEntry	Security
IADsAccessControlList	Security
IADsAcl	Data Type
IADsADSystemInfo	Utility
IADsAggregatee	Obsoleta
IADsAggregator	Obsoleta
IADsBackLink	Data Type
IADsCaseIgnoreList	Data Type
IADsClass	Schema
IADsCollection	Persistent object
IADsComputer	Persistent object
IADsComputerOperations	Dynamic object
IADsContainer	Core
IADsDeleteOps	Utility
IADsDNWithBinary	Data Type
IADsDNWithString	Data Type
IADsDomain	Persistent object
IADsEmail	Data Type
IADsExtension	Extension
IADsFaxNumber	Data Type
IADsFileService	Persistent object
IADsFileServiceOperations	Dynamic object
IADsFileShare	Persistent object
IADsGroup	Persistent object
IADsHold	Data Type

Interfaccia	Categoria
IADsLargeInteger	Data Type
IADsLocality	Persistent object
IADsMembers	Persistent object
IADsNamespaces	Core
IADsNameTranslate	Utility
IADsNetAddress	Data Type
IADsObjectOptions	Utility
IADsOctetList	Data Type
IADsOpenDSObject	Core
IADsOU	Persistent object
IADsPath	Data Type
IADsPathName	Utility
IADsPostalAddress	Data Type
IADsPrintJob	Persistent object
IADsPrintJobOperations	Dynamic object
IADsPrintQueue	Persistent object
IADsPrintQueueOperations	Dynamic object
IADsProperty	Schema
IADsPropertyEntry	Property Cache
IADsPropertyList	Property Cache
IADsPropertyValue	Property Cache
IADsPropertyValue2	Property Cache
IADsReplicaPointer	Data Type
IADsResource	Dynamic object
IADsSecurityDescriptor	Security
IADsService	Persistent object
IADsServiceOperations	Dynamic object
IADsSession	Dynamic object
IADsSyntax	Schema
IADsTimestamp	Data Type
IADsTsUserEx	Terminal Services user data
IADsTypedName	Data Type
IADsUser	Persistent object

Interfaccia	Categoria
IADsWinNTSystemInfo	Utility
IDirectoryObject	Core/Non-Automation
IDirectorySchemaMgmt	Obsoleta
IDirectorySearch	Core/Non-Automation
IPrivateDispatch	Obsoleta
IPrivateUnknown	Obsoleta

Tabella 2: Interfacce ADSI

4.10 IL BINDING AL GLOBAL CATALOG

Come abbiamo detto nel Capitolo 3, il Global Catalog possiede tutti gli oggetti della foresta, conservando però un insieme limitato degli attributi.
Tuttavia, essendo sempre un Domain Controller (il primo DC installato diventa automaticamente un Global Catalog), possiede anch'esso le tre partizioni di default previste da Active Directory e, quindi, per "distinguere" le richieste effettuate verso il suo catalogo globale, destina un'apposita porta TCP/IP (la 3268 per inciso) per questo genere di query.
Per poter interrogare un GC, le regole finora viste sono le stesse, ma cambia leggermente la stringa AdsPath che identifica il provider, passando da LDAP a GC. Quindi, stringhe come:

```
…
 GC:
 GC://RootDSE
 GC://MyGC/RootDSE
…
```

sono valide, anche se per diverse ragioni è preferibile servirsi del server-less binding.

4.11 IL BINDING TRAMITE GUID

Esiste un altro modo per indirizzare le nostre ricerche verso un particolare oggetto ed è quello di utilizzare il suo GUID. Ogni oggetto di Active Directory ha un GUID univoco, un valore a 128 bit generata dal sistema che rimane invariato e legato all'oggetto anche se questo viene rinominato o spostato. Questo valore è memorizzato nell'attributo objectGUID dell'oggetto e lo possiamo visualizzare, ad esempio, aprendo ADSIEdit e visualizzando le proprietà dell'oggetto desiderato. Per utilizzare questo tipo di binding, dobbiamo modificare la forma della nostra stringa AdsPath in questo modo:

```
LDAP://<GUID=GUID dell'oggetto>
```

Facciamo un esempio. Supponiamo di voler vedere alcune proprietà dell'utente riportato in figura 4, dalla figura si evince quindi che il GUID dell'utente è *0x4f 0x18 0x32 0xcd 0xd4 0x91 0x81 0x48 0xa0 0x6e 0x9e 0x83 0x22 0xd8 0x52 0x9e*.

CN=Francesco Dr. Lippo Properties
Attributes
Path: GC://server.MYDOMAIN.IT/CN=Francesco Dr. Lippo,CN=Users,
Class: user
Select which properties to view: Optional
Select a property to view: objectGUID
Attribute Values
Syntax: OctetString
Edit Attribute:
Value(s): 0x4f 0x18 0x32 0xcd 0xd4 0x91 0x81 0x48 0xa0 0x6e
Set Clear
OK Cancel Apply

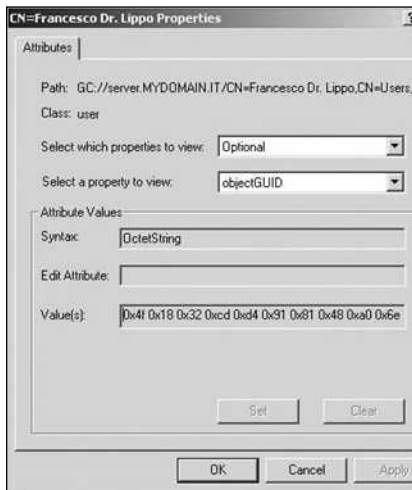



Figura 4: L'attributo objectGUID

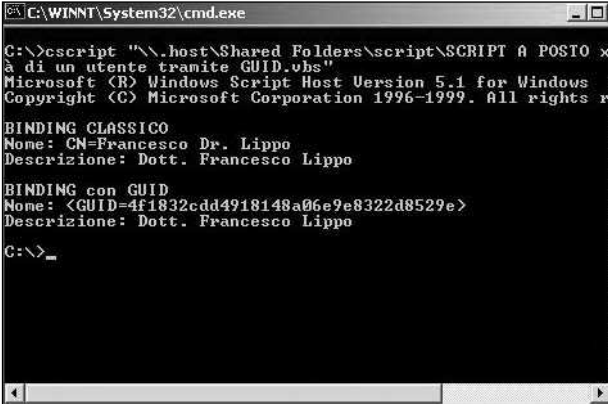
Riportando il tutto sottoforma di codice e tralasciando per il momento le istruzioni non ancora menzionate, vediamo in che modo il binding “classico” differisce da quello mediante GUID:

LISTATO 1 (Binding classico e mediante GUID)

```
Option Explicit
Dim objDSE
Dim objUsr
' Metodo 1

Set objDSE = GetObject
("LDAP://CN=Francesco Dr. Lippo,CN=Users,
DC=MyDomain,DC=it")
objDSE.GetInfo
Wscript.Echo "BINDING CLASSICO"
Wscript.Echo "Nome: " & objDSE.Name
Wscript.Echo "Descrizione: "

& objDSE.Description & vbCrlf
' Metodo 2

Set objUsr =
GetObject("LDAP://<GUID=4f1832cdd4918148a06e9e8322d8529e>")
objUsr.GetInfo
Wscript.Echo "BINDING con GUID"
Wscript.Echo "Nome: " & objUsr.Name
Wscript.Echo "Descrizione: "
& objUsr.Description
Set objDSE = Nothing
Set objUsr = Nothing
```

che a video produce un output del tipo:

```
C:\WINNT\System32\cmd.exe

C:\>cscript "\\.host\Shared Folders\script\SCRIPT A POSTO x
à di un utente tramite GUID.vbs"
Microsoft (R) Windows Script Host Version 5.1 for Windows
Copyright (C) Microsoft Corporation 1996-1999. All rights r

BINDING CLASSICO
Nome: CN=Francesco Dr. Lippo
Descrizione: Dott. Francesco Lippo

BINDING con GUID
Nome: <GUID=4f1832cdd4918148a06e9e8322d8529e>
Descrizione: Dott. Francesco Lippo

C:\>_
```

Figura 5: Binding ad un oggetto tramite GUID

Si noti la differenza di output dei due metodi e soprattutto il fatto che l'attributo Name dell'oggetto è diverso nei due casi.

4.12 WELL KNOWN OBJECT BINDING

Riguardo all'operazione di binding, ci sarebbe ancora molto da dire e da sottolineare. Una di queste riguarda i cosiddetti Well Known Object. Facciamo un piccolo passo indietro e ripensiamo al binding effettuato attraverso il GUID di un oggetto.
Sappiamo che un GUID è creato dal sistema nel momento in cui un oggetto viene creato e rimane invariato per tutta la vita di quell'oggetto. Il GUID binding, come appena visto, presuppone la conoscenza di questa sequenza di bit affinchè possa essere utilizzato. Tuttavia, esistono all'interno di Active Directory, alcuni oggetti ai quali viene assegnato un GUID predefinito attraverso il quale possiamo compiere moltissime operazioni. Questi oggetti sono proprio i Well Known Object, argomento di questo paragrafo.
I Well Known Object sono sempre e soltanto contenuti all'interno di

ogni Domain e Configuration Container. In particolare, ecco la lista di questi oggetti:

Domain Container:

- Users
- Computers
- System .
- Domain Controllers
- Infrastructure
- Deleted Objects
- Lost and Found

Configuration:

- Deleted Objects

Per effettuare il binding a ciascuno di essi, è necessario utilizzare la sintassi

```
LDAP://servername/<WKGUID=XXXXXXXXXXXXXXXX,ContainerDN>
```

specificando il GUID dell'oggetto che si desidera ed il container di riferimento. Ecco la lista delle costanti che che sfrutteremo e che identificano ciascun oggetto:

Container	Costante GUID	Valore
Users	GUID_USERS_CONTAINER	a9d1ca15768811d1aded 00c04fd8d5cd
Computers	GUID_COMPUTRS_CONTAINER	aa312825768811d1aded 00c04fd8d5cd
System	GUID_SYSTEMS_CONTAINER	ab1d30f3768811d1aded 00c04fd8d5cd

Container	Costante GUID	Valore
Domain Controllers	GUID_DOMAIN_ CONTROLLERS_CONTAINER	A361b2ffffd211d1aa4b00 c04fd7d83a
Infrastructure	GUID_INFRASTRUCTURE_ CONTAINER	2fbac1870ade11d297c400 c04fd8d5cd
Deleted Objects	GUID_DELETED_OBJECTS_ CONTAINER	18e2ea80684f11d2b9aa00 c04f79f805
Lost and Found	GUID_LOSTANDFOUND_ CONTAINER	ab8153b7768811d1aded00 c04fd8d5cd

Tabella 3: I Well Known Objects

Prima di mostrare un esempio sull'utilizzo di questi oggetti, è importante sottolineare che:

- il valore GUID a fianco di ciascun container non rispecchia il valore dell'attributo objectGUID visto in precedenza;
- l'utilizzo di questa notazione (la WKGUID) è supportata esclusivamente da Active Directory.

Ecco un esempio di script che utilizza parzialmente i Well Known Object sul dominio MyDomain.it:

LISTATO 2 (Binding con i Well Known GUID)

```
Option Explicit
Const GUID_USERS_CONTAINER =
"a9d1ca15768811d1aded00c04fd8d5cd"
Const GUID_COMPUTRS_CONTAINER =
"aa312825768811d1aded00c04fd8d5cd"
Const GUID_SYSTEMS_CONTAINER =
"ab1d30f3768811d1aded00c04fd8d5cd"
Const GUID_DOMAIN_CONTROLLERS_CONTAINER =
"a361b2ffffd211d1aa4b00c04fd7d83a"
Const GUID_INFRASTRUCTURE_CONTAINER =
```

```
"2fbac1870ade11d297c400c04fd8d5cd"
Const GUID_DELETED_OBJECTS_CONTAINER =
"18e2ea80684f11d2b9aa00c04f79f805"
Const GUID_LOSTANDFOUND_CONTAINER =
"ab8153b7768811d1aded00c04fd8d5cd"
Dim objContainer
Dim objChild
Binding a GUID_USERS_CONTAINER e
GUID_COMPUTRS_CONTAINER
Set objContainer =
GetObject("LDAP://<WKGUID=" & GUID_USERS_CON
TAINER & ",DC=MyDomain,DC=it>")
Wscript.Echo "GUID_USERS_CONTAINER"
& vbCrlf &

"________________________"
For Each objChild In objContainer
  WScript.Echo objChild.Name
Next
Wscript.Echo "--------------------------------"
Set objContainer =
GetObject("LDAP://<WKGUID=" & GUID_COMPU
TRS_CONTAINER & ",DC=MyDomain,DC=it>")
Wscript.Echo
"GUID_COMPUTRS_CONTAINER" & vbCrlf &
"________________________"
For Each objChild In objContainer
Per ogni oggetto trovato,
stampa il nome
  WScript.Echo objChild.Name
Next
Set objContainer = Nothing
Set objChild = Nothing
```

```
C:\WINNT\System32\cmd.exe
11-47 ADSI Bind to a WKGUID.vbs"
Microsoft (R) Windows Script Host Version 5.1 for Wind
Copyright (C) Microsoft Corporation 1996-1999. All rig

GUID_USERS_CONTAINER
_____________________
CN=Administrator
CN=Cert Publishers
CN=DnsAdmins
CN=DnsUpdateProxy
CN=Domain Admins
CN=Domain Computers
CN=Domain Controllers
CN=Domain Guests
CN=Domain Users
CN=Enterprise Admins
CN=Francesco Dr. Lippo
CN=Group Policy Creator Owners
CN=GrpLocalTest
CN=GrpTest1
CN=GrpTest2
CN=GruppoDiTest
CN=Guest
CN=IUSR_SERVER
CN=IWAM_SERVER
CN=krbtgt
CN=MYDOMAIN2$
CN=RAS and IAS Servers
CN=Schema Admins
CN=Test
CN=TsInternetUser
---------------------------------
GUID_COMPUTRS_CONTAINER
_____________________
CN=COMPUTER1
CN=COMPUTER2
CN=COMPUTER3

C:\>_
```

Figura 6: Output dello script che utilizza i WKGUID

MANIPOLARE GLI OGGETTI DI AD

Active Directory è composta da centinaia di oggetti.
Naturalmente, l'operazione più importante per chi programma, è quella d'interrogare o impostare le proprietà degli oggetti di quest'enorme "database" per ottenere informazioni di qualunque genere.
In questo capitolo cercheremo di affrontare questo problema mostrando i passi necessari per recuperare correttamente ciò che ci serve e lo faremo partendo dall'interfaccia ADSI forse più importante: la IADs.

Premessa

Abbiamo visto nel capitolo precedente come effettuare il binding ad un oggetto qualunque di Active Directory, ma abbiamo tralasciato volutamente di considerare la parte di codice che leggeva e/o impostava eventuali attributi.
Adesso è arrivato il momento di vedere come leggerli e soprattutto come modificarli.
Partiremo dall'interfaccia COM che può essere ritenuta quella principale, dalla quale in qualche modo, vengono poi specializzate le altre: la IADs. Vedremo che una volta appresi i concetti base, la programmazione risulterà immediata

5.1 L'INTERFACCIA IADS

Ogni oggetto COM che fa parte della "suite" messa a disposizione da ADSI, in quanto tale, deve necessariamente supportare l'interfaccia IUnknown.
Oltre a questa, ognuno di essi supporta l'interfaccia IADs che, come avremo modo di vedere, ci tornerà molto utile per recuperare e impostare i valori degli attributi desiderati
Ecco innanzitutto l'elenco delle proprietà e dei metodi che caratterizzano questa interfaccia:

Proprietà	Data Type	Descrizione
ADsPath	String	ADsPath dell'oggetto
Class	String	Il nome della classe a cui appartiene l'oggetto
GUID	String	Il GUID dell'oggetto
Name	String	L'RDN dell'oggetto
Parent	String	ADsPath dell'oggetto genitore
Schema	String	ADsPath della classe (relativa all'oggetto) definite all'interno dello schema

Tabella 1: Proprietà dell'interfaccia IADs

Metodo	Descrizione
GetInfo	Recupera tutti gli attributi dell'oggetto copiandoli nella local property cache
SetInfo	Salva i cambiamenti dell'oggetto all'interno della directory
Get	Recupera il valore di un attributo specifico
Put	Imposta il valore di un attributo specifico
GetEx	Recupera il valore o i valori di un attributo specifico, ritornando un array
PutEx	Consente di modificare, cancellare, ecc. i valori di uno o più attributi specificati
GetInfoEx	Recupera il valore di un'attributo specifico aggiornando la local property cache

Tabella 2: I Well Known Objects

Dalle due tabelle si evince chiaramente che se vogliamo leggere il valore di un qualunque attributo di un oggetto, dovremo usare i metodi Get e "simili", mentre per modificarli, dovremo avvalerci dei metodi di Put.
Prima di vedere in dettaglio i metodi principali di quest'interfaccia, vediamo un piccolo esempio che ci mostra come recuperare i valori

delle proprietà dell'oggetto COM relativi all'interfaccia IADs:

LISTATO 2 (Proprietà di un oggetto (IADs Interface))

```
Option Explicit
Dim objIAD
Dim strLDAP
strLDAP = "LDAP://CN=Administrator,CN=Users,DC=MyDomain,DC=IT"

Set objIAD = GetObject(strLDAP)

Wscript.Echo "-----------------------------------"
Wscript.Echo "- ADMINISTRATOR (Proprietà IADs)    "
Wscript.Echo "-----------------------------------"
WScript.Echo "AdsPath:"  & objIAD.AdsPath
WScript.Echo "Class  :"  & objIAD.Class
WScript.Echo "GUID   :<"  & objIAD.GUID & ">"
WScript.Echo "Name   :"  & objIAD.Name
WScript.Echo "Parent :"  & objIAD.Parent
WScript.Echo "Schema :"  & objIAD.Schema
```

che produce qualcosa del tipo:

```
-----------------------------------------------
- ADMINISTRATOR (Proprietà IADs)
-----------------------------------------------
AdsPath      :
LDAP://CN=Administrator,CN=Users,DC=MyDomain,DC=IT
Class        : user
GUID         :<75b537243ada644cba795df0517bb51e>
Name         :CN=Administrator
Parent       :LDAP://CN=Users,DC=MyDomain,DC=IT
Schema       :LDAP://schema/user
```

Come avremo certamente notato, una volta effettuato il binding all'oggetto desiderato, è stato sufficiente sfruttare la notazione oggetto.proprietà per ottenere il valore della proprietà desiderata. Ma esiste ancora qualche considerazione da fare.

5.2 IL METODO GET

Uno dei metodi utilizzati per leggere i valori degli attributi di un oggetto Active Directory è quello di utilizzare il metodo Get, specificando il nome dell'attributo desiderato. Ad esempio, considerando che la classe User, che descrive ogni utente all'interno di Active Directory, possiede la proprietà Description (che riflette il campo Descrizione presente nel pannello Generale delle proprietà di ogni utente), proviamo ad inserire, alla fine dello script precedente, le seguenti righe:

```
objUserDescr = objUser.Description
Wscript.Echo "1) " & objUserDescr
objUserDescr = objUser.Get("Description")
Wscript.Echo "2) " & objUserDescr
```

Se confrontiamo l'output di entrambe, noteremo facilmente che producono lo stesso risultato. Adesso, sostituiamole con:

```
objUserDescr = objUser.Name
Wscript.Echo "1) " & objUserDescr
objUserDescr = objUser.Get("Name")
Wscript.Echo "2) " & objUserDescr
```

Con nostra meraviglia ci renderemo presto conto che non producono più lo stesso risultato, ma qualcosa del tipo:

```
1) CN=Administrator
```

```
2) Administrator
```

Facciamo un ulteriore test. Consideriamo, al posto della proprietà Name, la proprietà GUID e aggiungiamo in coda allo script precedente, analogamente a prima, le seguenti righe di codice:

```
objUserGUID = objUser.GUID
Wscript.Echo "1) " & objUserGUID
objUserGUID = objUser.Get("GUID")
Wscript.Echo "2) " & objUserGUID
```

Se lanciamo lo script, le prime due istruzioni non daranno problemi, mentre la terza produrrà un errore.
A questo punto è arrivato il momento di spiegare dove e perché si riscontrino questi problemi e lo faremo considerando soltanto quest'ultime righe di codice. La prima istruzione non fallisce perché, così com'è presentata, si riferisce alla proprietà GUID dell'oggetto COM/ADSI visto in Tabella 4. La terza istruzione, invece, tenta di recuperare il valore di un attributo di un utente, presente in Active Directory, inesistente. Infatti, la proprietà GUID di un utente non è mappata all'interno di AD con il nome GUID, bensì come objectGUID.Questo spiega anche il perché, malgrado i risultati fossero leggermente differenti (a meno del prefisso CN=), con la proprietà Name le cose andavano meglio. Ecco l'elenco delle corrispondenze:

IADs Property	LDAP Attribute
AdsPath	DistinguishedName
Class	ObjectClass
GUID	ObjectGUID
Name	-
Parent	-
Schema	ObjectCategory

Tabella 3: Corrispondenza LDAP Attribute - ADSI Property

In definitiva, quindi, occorre fare attenzione alla differenza che esiste tra gli attributi LDAP (objectGUID ad esempio) e le proprietà di un oggetto ADSI (GUID nel caso specifico). Inoltre, abbiamo visto che non tutti gli attributi sono recuperabili attraverso la sintassi oggetto.proprietà (o comunque non sempre la corrispondenza dei nomi è la stessa), mentre qualunque attributo può essere recuperato attraverso il metodo Get.

5.3 ATTRIBUTI MULTIVALORI: IL METODO GETEX

Gli attributi menzionati in precedenza sono piuttosto semplici da considerare. Basta semplicemente conoscere il loro nome LDAP ed utilizzare il metodo Get dell'interfaccia IADs per leggerli. Ma esistono attributi che non memorizzano sempre un solo valore, ma possono contenerne più di uno. Immaginiamo, ad esempio, l'attributo che memorizza la lista dei membri di un determinato gruppo di Windows 2000.
Questo attributo, denominato member, può contenere anche più di un valore ed il metodo Get visto in precedenza, è ovvio, non può più essere utilizzato. In casi come questo ci viene incontro il metodo GetEx, molto simili al precedente, ma in grado di ritornare un array di Variant che possiamo scorrere facilmente.
A questo proposito, prendiamo in considerazione lo script seguente:

LISTATO 3 (Lista dei membri del gruppo GrpTest1)

```
Option Explicit

Dim objGroup
Dim arrMembers
Dim Member
Set objGroup =
```

```
GetObject
("LDAP://cn=GrpTest1,cn=Users,dc=MyDomain,dc=it")
objGroup.GetInfo

arrMembers = objGroup.GetEx("member")

WScript.Echo "Members:"
Wscript.Echo "LISTA DEI MEMBRI DEL GRUPPO GRPTEST1"
Wscript.Echo
"-----------------------------------"
For Each Member In arrMembers
   WScript.Echo Member
Next
```

Questo esempio mostra a video la lista di tutti i membri di un gruppo chiamato GrpTest1, contenuto all'interno del dominio MyDomain.it. L'output a video sarà qualcosa del tipo:

Figura 1: Output dello script d'esempio

Solo per inciso, è bene tener presente che per potersi accertare del risultato, ma soprattutto per poter fare un pò di pratica con questi oggetti, possiamo verificare quanto ci viene riportato dallo script utilizzando il tool ADSIEdit.

Ecco, infatti, quanto ci viene ritornato da questo tool in corrispondenza dell'attributo member dell'oggetto GrpTest1:

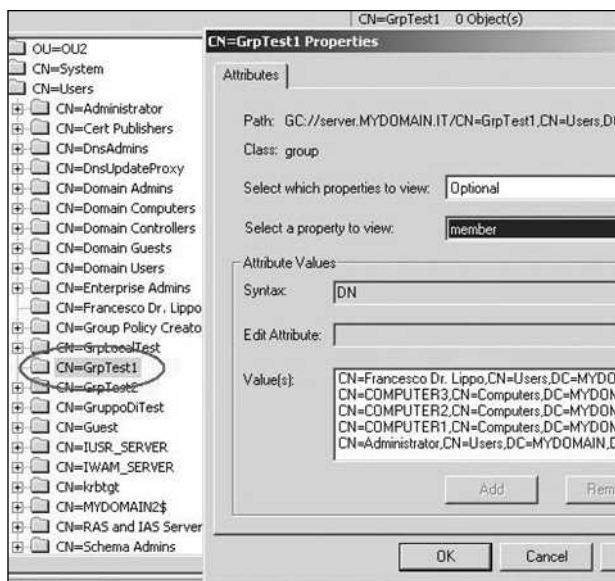



Figura 2: Valore dell'attributo "member" dell'oggetto GrpTest1

5.4 IL METODO PUT

Se ripensiamo alle considerazioni fatte per i metodi Get e GetEx, non dovrebbe essere complicato comprendere i metodi che stiamo per vedere. Consideriamo, ancora una volta, il caso dell'utente Administrator e della proprietà (attributo) Description.
Ecco quindi un semplice script che consente di modificarlo in maniera molto semplice:

LISTATO 4 (Modifica dell'attributo Description)

```
Option Explicit
Dim objUsr
Dim strLDAP
```

```
strLDAP =
"LDAP://CN=Administrator,CN=Users,DC=MyDomain,DC=IT"

Set objUsr = GetObject(strLDAP)

Wscript.Echo "--------------------------------------------------------------"
Wscript.Echo
"Modifica della descrizione dell'utente Administrator"
Wscript.Echo "--------------------------------------------------------------"
objUsr.Description = "Questa è una descrizione nuova!"
objUsr.SetInfo
```

Naturalmente, così come spiegato nel paragrafo precedente, se l'oggetto ADSI non supporta quel determinato attributo, è necessario ricorrere, in maniera analoga, al metodo Put.
Quindi, la penultima istruzione andrebbe modificata in questo modo:

```
objUsr.Put "Description", "Questa è una descrizione nuova!"
```


5.5 LA LOCAL PROPERTY CACHE

Molto probabilmente qualcuno si sarà accorto che, all'interno della tabella relativa ai metodi dell'interfaccia IADs, esistono alcuni riferimenti alla cosiddetta Property Cache.
Se riflettiamo un attimo sulla complessità di Active Directory, ci rendiamo conto che, all'interno di un'architettura nella quale esistono molteplici server, sparsi magari su sedi diverse, non ha molto senso "costringere" ADSI ad interrogare di volta in volta il database di AD, soprattutto quando le interrogazioni riguardano sempre o spesso, gli stessi oggetti.Per queste ragioni, ADSI conserva, lato client, una cache locale denominata, per l'appunto, Property Cache il cui funzionamento è schematizzato di seguito:

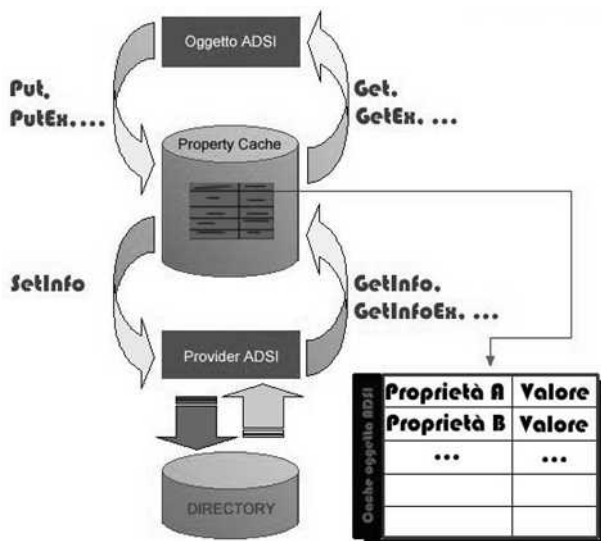



Figura 3: La Property Cache

La Property Cache è creata in locale al momento in cui effettuiamo il binding ad un oggetto, ma rimane vuota fino a quando non viene realmente fatta richiesta di ottenere il valore di una o più proprietà. In quel caso, ADSI riempie questa struttura con diverse informazioni, tra cui il nome delle proprietà ed il valore/valori dell'oggetto richiesto. In questo modo, quindi, le successive query saranno soddisfatte molto più velocemente perché ADSI, prima di "rivolgersi" direttamente ad Active Directory, controllerà la cache alla ricerca di quell'attributo.

5.6 GETINFO, GETINFOEX E SETINFO

Attorno al funzionamento della Property Cache esistono diverse considerazioni che si possono fare. Ad esempio: come possiamo essere sicuri che i dati che preleviamo siano quelli reali? In fondo, se non uti-

lizziamo i nostri script per molto tempo, potrebbe essere accaduto che dati contenuti nella Property Cache siano cambiati.
E ancora: per quale ragione dobbiamo necessariamente riempire questa tabella con "tutti" gli attributi di un oggetto, quando magari a noi ne interessa solo qualcuno? Bene, queste considerazioni, del tutto lecite, trovano risposta nei metodi GetInfo e GetInfoEx dell'oggetto IADs.
Con il primo metodo, infatti, forziamo l'aggiornamento della cache locale, assicurandoci quindi che i dati in essa contenuti siano sempre quelli attuali. La sintassi è piuttosto semplice e la si evince dai listati precedenti:

```
Oggetto.GetInfo
```

Se, inoltre, non volessimo aggiornarla con tutti gli attributi dell'oggetto, anche e soprattutto per motivi di performance, possiamo affidarci al metodo GetInfoEx la cui sintassi è:

```
Oggetto.GetInfoEx <Array attributi>, 0
```

Ecco un frammento d'esempio che permette di caricare soltanto gli attributi "cn" e "member" del gruppo GrpTest1:

```
Set objGroup = GetObject("LDAP://CN=GrpTest1,CN=Users,DC=
MYDOMAIN,DC=IT")
objGroup.GetInfoEx Array("cn","member"), 0
```

Per ultimo, ma sicuramente non per importanza, va sottolineato il metodo SetInfo.
Questo metodo (lo si evince anche dalla schematizzazione del funzionamento della Property Cache precedente) effettua il vero e proprio update delle modifiche apportate agli attributi degli oggetti contenuti nella cache, all'interno di Active Directory.

Sino a quando non viene portata a termine questa operazione, tutte le informazioni sino a quel momento modificate, rimangono all'interno della Property Cache e questo, ovviamente, anche per una maggiore sicurezza (oltre che per ragioni di performance).
Inoltre, giusto per inciso, è interessante sapere che al momento in cui viene lanciata questa "commit" verso la directory, non tutti gli attributi diventano oggetto di quest'operazione, ma soltanto quelli realmente modificati (durante le varie operazioni, infatti, ogni oggetto che subisce una qualunque modifica, viene marcato opportunamente in maniera tale da essere riconosciuto al momento in cui si lancia il metodo SetInfo).

5.7 IL METODO PUTEX

Abbiamo visto come leggere attributi singoli (metodo Get) e multipli (metodo GetEx) e come impostare i valori di attributi a singolo valore (metodo Put).
Analogamente a quanto evidenziato per il metodo GetEx, ci si potrebbe aspettare che esista un metodo analogo ad esso, ma in grado di aggiornare attributi a valore multiplo.
Ovviamente, questo metodo esiste, ma diversamente dal suo "omonimo" GetEx è molto più potente e molto più flessibile dell'analogo metodo Put. Stiamo parlando del metodo PutEx.
Cominciamo vedendo la tipica sintassi di un'istruzione che utilizza il metodo PutEx:

```
Oggetto.PutEx <Control Code>, <Attributo>, <Valori>
```

Si osservi per prima cosa la presenza di un parametro aggiuntivo (il primo per l'esattezza) rispetto al metodo Put.
Questo parametro, definito attraverso delle costanti, definisce il tipo di operazione che il metodo deve compiere, secondo la seguente tabella:

Property Control Code	Valore	Descrizione
ADS_PROPERTY_CLEAR	1	Cancella tutti i valori di un attributo
ADS_PROPERTY_UPDATE	2	Aggiorna i valori di un attributo con i nuovi
ADS_PROPERTY_APPEND	3	Aggiunge i valori passati come parametro a quelli presenti
ADS_PROPERTY_DELETE	4	Rimuove un determinato valore da un'attributo

Tabella 4: Le costanti (Control Code) relative al metodo PutEx

Vediamo qualche eseFmpio che ci aiuti a comprendere meglio l'utilizzo di questo metodo.

LISTATO 5 (Modifica dei membri di un gruppo con PutEx)

```
Option Explicit
Const ADS_PROPERTY_CLEAR    = 1
Const ADS_PROPERTY_UPDATE   = 2
Const ADS_PROPERTY_APPEND   = 3
Const ADS_PROPERTY_DELETE   = 4
Dim objGroup
Set objGroup =
GetObject("LDAP://CN=GrpTest1,CN=Users,
DC=MYDOMAIN,DC=IT")
'--------------------------------------------------------------------------
' Commentare le istruzioni con PutEx che non si desidera sfruttare
'--------------------------------------------------------------------------
' CLEAR - Cancella tutto il contenuto
 objGroup.PutEx ADS_PROPERTY_CLEAR, "member", vbNullstring
' UPDATE - Aggiorna con Utente1 e Utente2
```

```
objGroup.PutEx ADS_PROPERTY_UPDATE, "member",
Array("CN=Utente1,CN=Users,DC=MYDOMAIN,DC=IT",_
        "CN=Utente2,CN=Users,DC=MYDO
MAIN,DC=IT")

' APPEND - Aggiunta Utente1 e Utente2
objGroup.PutEx ADS_PROPERTY_APPEND, "member",
Array("CN=Utente1,CN=Users,DC=MYDOMAIN,DC=IT",_
  "CN=Utente2,CN=Users,DC=MYDO
MAIN,DC=IT")

' DELETE - Eliminazione Utente1
objGroup.PutEx ADS_PROPERTY_DELETE, "member",
Array("CN=Utente1,CN=Users,DC=MYDOMAIN,DC=IT")

objGroup.SetInfo
```

5.8 I CONTAINERS

L'argomento che stiamo per trattare probabilmente doveva trovare posto molto prima di questo capitolo o quantomeno all'inizio della seconda parte del libro. Questo perché Active Directory, se lo vediamo da un punto di vista soprattutto "astratto", è composto da diversi oggetti, la maggior parte dei quali sono soprattutto container.
In effetti, abbiamo visto, parlando della struttura di un generico albero LDAP che rappresentava Active Directory, che ogni nodo di questa struttura era quasi sempre rappresentato da oggetti che ne contenevano altri. Durante il cammino che ci ha portato sin qui, abbiamo visto o parlato (magari non rendendocene conto) di questo tipo di oggetti (i gruppi, ad esempio o le Organizational Unit).
Per poter interagire con questo genere di oggetti, ADSI mette a disposizione del programmatore un'apposita interfaccia, definita tra le Core Interfaces e denominata IADsContainer.

Immaginiamo per un attimo di considerare il seguente contenitore di AD:

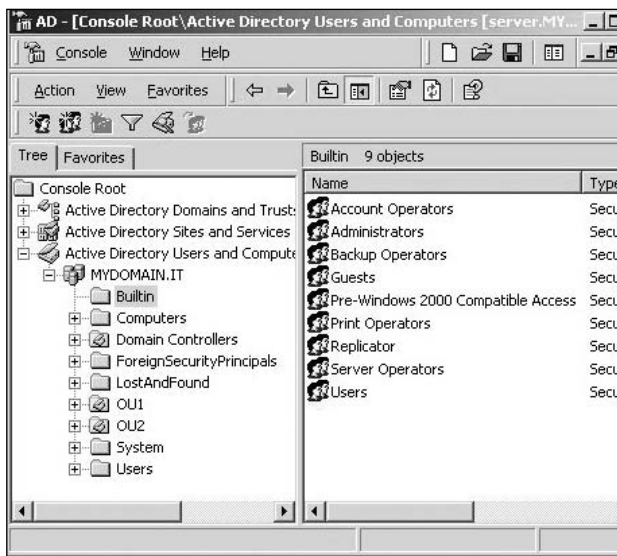



Figura 4: I membri del container Builtin

Basandoci anche sugli script precedenti, non dovrebbe essere complicato comprendere che il seguente listato consente di ottenere la lista di tutti gli "oggetti" del contenitore Builtin:

```
Option Explicit
Dim objDSE
Dim strLDAP
Dim Object
Dim objDomain
Set objDSE = GetObject("LDAP://rootDSE")
strLDAP = "LDAP://cn=Builtin," & objDSE.Get("defaultNamingContext")
Set objDomain = GetObject(strLDAP)
```

```
Call ListObject(objDomain)
Sub ListObject(objContainer)
  For Each Object In objContainer
  Wscript.Echo Object.Name
  Next
End Sub
```

LISTATO 6 Elenca tutti i membri del container "Builtin"

Questa operazione, quella di enumerazione degli elementi del contenitore Builtin è resa possibile grazie all'intervento dell'interfaccia IADsContainer. Questa interfaccia è fondamentale perché è praticamente impossibile intervenire su oggetti di Active Directory senza tenerne conto. Di seguito mostriamo i metodi e le proprietà supportati:

Metodo	Descrizione
get_Count	Ritorna il numero di oggetti del container (non implementato sotto AD).
get__NewEnum	Ritorna un oggetto enumerator per il contenitore.
get_Filter	Ritorna il filtro sugli elementi che verranno enumerati (sulle classi dello schema).
put_Filter	Imposta un filtro sugli elementi che verranno enumerati (sulle classi dello schema).
get_Hints	Ritorna le proprietà che saranno recuperate per ciascun oggetto del container.
put_Hints	Imposta le proprietà che saranno recuperate per ciascun oggetto del container.
GetObject	Ritorna l'interfaccia per un oggetto del container.
Create	Crea un nuovo oggetto nel container.
Delete	Elimina un oggetto del container.
CopyHere	Copia un oggetto del container.
MoveHere	Sposta un oggetto del container.

Tabella 5: Metodi dell'interfaccia IADsContainer

Proprietà	Descrizione
Count	Identifica il numero di oggetti del container.
Filter	Filtro impostato per enumerare gli oggetti.
Hints	Contiene le proprietà degli item da enumerare contenuti nel container.

Tabella 6: Proprietà dell'interfaccia IADsContainer

Facciamo vedere ancora un esempio che consente filtrare il contenuto degli oggetti del container che vengono enumerati. In questo caso prendiamo come riferimento il contenitore Users:

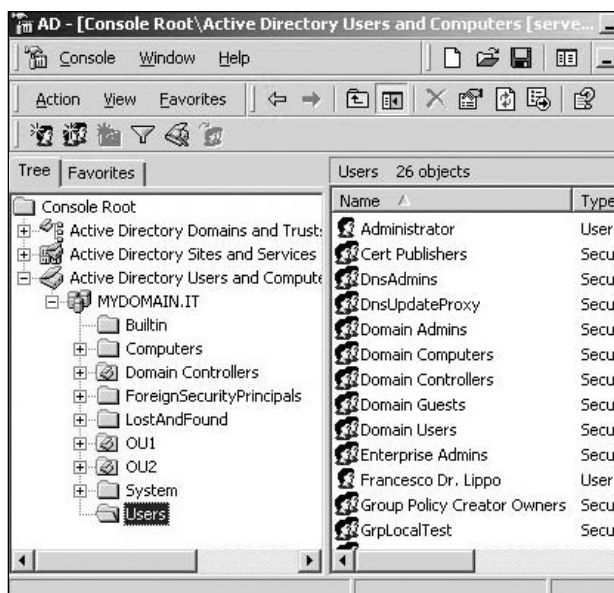



Figura 5: Un tipico errore non gestito dall'applicazione.

Attraverso uno script simile al precedente, ma con l'aggiunta di qualche istruzione che consente di filtrare gli oggetti del contenitore in base alla classe a cui appartengono, potremo selezionare

i soli oggetti di tipo “gruppo”:

```
Option Explicit
Dim objDSE
Dim strLDAP
Dim Object
Dim objDomain
Set objDSE = GetObject("LDAP://rootDSE")
strLDAP = "LDAP://cn=Users," & objDSE.Get
("defaultNamingContext")
Set objDomain = GetObject(strLDAP)
Call ListObject(objDomain)
Sub ListObject(objContainer)
  ' CON FILTRO
    WScript.Echo "LISTA FILTRATA SU group"
    objContainer.Filter = Array("group")
    For Each Object In objContainer
        Wscript.Echo Object.Name &
" Class: " & Object.Class
    Next
        ' SENZA FILTRO  WScript.Echo
"LISTA SENZA FILTRO"
   objContainer.Filter = Nothing
   For Each Object In objContainer
       Wscript.Echo Object.Name &
" Class: " & Object.Class
   Next
End Sub
```

LISTATO 7 Lista degli elementi del container Users con e senza Filter

Questo listato produce un risultato simile a:

LISTA FILTRATA SU group

```
CN=Cert Publishers Class: group
CN=DnsAdmins Class: group
CN=DnsUpdateProxy Class: group
```

LISTA SENZA FILTRO

```
CN=Administrator Class: user
CN=Cert Publishers Class: group
CN=DnsAdmins Class: group
CN=Francesco Dr. Lippo Class: user
CN=Group Policy Creator Owners Class: group
```

Nella seconda parte dell'output, si noti la presenza di oggetti appartenenti anche ad altre classi di oggetti, mentre nella prima erano visibili solo quelli di tipo "group". Inoltre, se volessimo migliorare il funzionamento di questo script, potremmo avvalerci di quest'istruzione, inserita prima dell'enumerazione:

```
objContainer.Hints = Array("Name", "Class")
```

che permette di utilizzare i soli attributi che Name e Class che c'interessano, migliorando le performance dello script.

LE PROPERTY CACHE INTERFACES

Quando dobbiamo recuperare i valori degli attributi di un determinato oggetto o dobbiamo modificarli, abbiamo visto quanto sia importante l'interfaccia IADs che, attraverso i suoi metodi, ci consente effettuare facilmente queste operazioni.
Abbiamo anche accennato all'esistenza di una Property Cache che conserva gli attributi degli oggetti in locale ma, a parte questo, è in-

discutibile che dobbiamo conoscere il nome di questi attributi prima di poterci mettere le mani sopra. Ovviamente la via più logica per scandire questa lista sarebbe quella d'interrogare direttamente lo Schema di Active Directory, ma vedremo che esistono alcune interfacce ADSI che possono in parte aiutarci e facilitarci il compito.

5.9 PROPRIETÀ DI UNA CLASSE

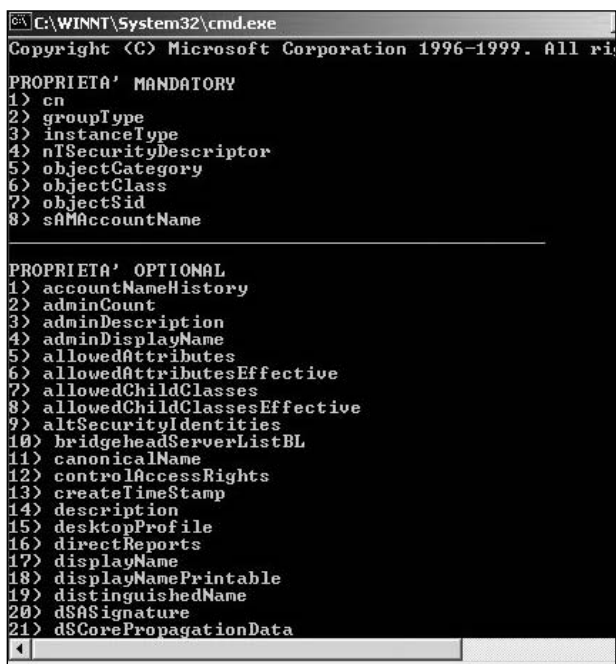
Abbiamo già accennato più volte al ruolo importante che ricopre lo Schema all'interno di Active Directory. Abbiamo anche accennato che al suo interno è contenuto l'elenco di tutte le classi di oggetti, con la definizione degli attributi e quant'altro sia necessario a gestirli correttamente. Appare quindi ovvio che, se volessimo elencare le proprietà di un determinato oggetto, dovremmo necessariamente sapere quale sia la classe alla quale l'oggetto stesso appartiene ed interrogare di conseguenza lo Schema alla ricerca di queste informazioni. Il primo passo per raggiungere il risultato è quello di ottenere l'informazione relativa allo Schema Container, che sappiamo essere replicato su ogni Domain Controller della Foresta. Per ottenere il suo Distinguished Name, abbiamo visto che è possibile sfruttare l'attributo schemaNamingContext dell'oggetto RootDSE attraverso il quale possiamo effettuare l'operazione di binding e recuperare infine le informazioni che ci servono. Tuttavia esiste un'altra possibilità, analoga come risultato alla precedente, ma che consente di effettuare il binding allo Schema in maniera molto più rapida, con meno istruzioni. Ecco la generica stringa utile da utilizzare per "agganciare" lo Schema di AD:

```
LDAP://schema
```

Se, in fondo a questa stringa aggiungiamo un "/" seguito dal nome della classe che vogliamo, avremo ottenuto un riferimento all'oggetto "classe" contenuto all'interno dello Schema e potremo leg-

gerne gli attributi.
Il listato seguente dovrebbe rendere la spiegazione precedente ancor più chiara:

```
Option Explicit
Dim objClass
Dim strPropName
Dim Cont
Set objClass = GetObject("LDAP://schema/group")
Cont = 0
Wscript.Echo "PROPRIETA' MANDATORY"
For Each strPropName In objClass.MandatoryProperties
    Cont = Cont + 1
    WScript.Echo Cont & ") " & strPropName
Next
Cont = 0
Wscript.Echo "PROPRIETA' OPTIONAL"
For Each strPropName In objClass.OptionalProperties
    Cont = Cont + 1
    WScript.Echo  Cont & ") " & strPropName
Next
```

LISTATO 8 Lista degli attributi della classe GROUP

In questo script, si elencano i nomi degli attributi della classe group ossia la generica classe che consente di descrivere i gruppi di Windows 2000. In particolare si osservi la presenza dell'attributo MandatoryProperties e dell'attributo OptionalProperties.
Queste due proprietà dell'oggetto che, all'interno dello schema, definisce la generica classe, costituiscono l'array delle proprietà obbligatorie e di quelle opzionali per quella classe. Avremo modo di incontrare in altri script questi due termini e soprattutto di comprenderne meglio l'importanza.

Tornando allo script appena mostrato, tutto quello che fa è:

- si connette allo Schema Container di Active Directory;
- effettua il binding alla classe group. Trattandosi di un oggetto "classe", l'oggetto restituito supporta l'interfaccia IADsClass che consente di manipolare proprio questi oggetti dello Schema;
- sfruttando le proprietà MandatoryProperties e OptionalProperties dell'oggetto IADsClass, elenca tutti i nomi degli attributi della classe group.

Ecco il possibile output del programma:

```
C:\WINNT\System32\cmd.exe
Copyright (C) Microsoft Corporation 1996-1999. All ri

PROPRIETA' MANDATORY
1) cn
2) groupType
3) instanceType
4) nTSecurityDescriptor
5) objectCategory
6) objectClass
7) objectSid
8) sAMAccountName
_______________________________________________

PROPRIETA' OPTIONAL
1) accountNameHistory
2) adminCount
3) adminDescription
4) adminDisplayName
5) allowedAttributes
6) allowedAttributesEffective
7) allowedChildClasses
8) allowedChildClassesEffective
9) altSecurityIdentities
10) bridgeheadServerListBL
11) canonicalName
12) controlAccessRights
13) createTimeStamp
14) description
15) desktopProfile
16) directReports
17) displayName
18) displayNamePrintable
19) distinguishedName
20) dSASignature
21) dSCorePropagationData
```

Figura 6: Output dello script che mostra gli attributi della classe group

In *Figura 6*, invece, troviamo le stesse informazioni viste attraverso l'interfaccia dello snap-in Active Directory Schema. Tuttavia, è bene sapere che per poter accedere ad informazioni di questo tipo, possiamo sfruttare alcuni importanti oggetti COM messi a disposizione da ADSI ed in grado d'interagire direttamente con le informazioni della Property Cache. Vediamo come.

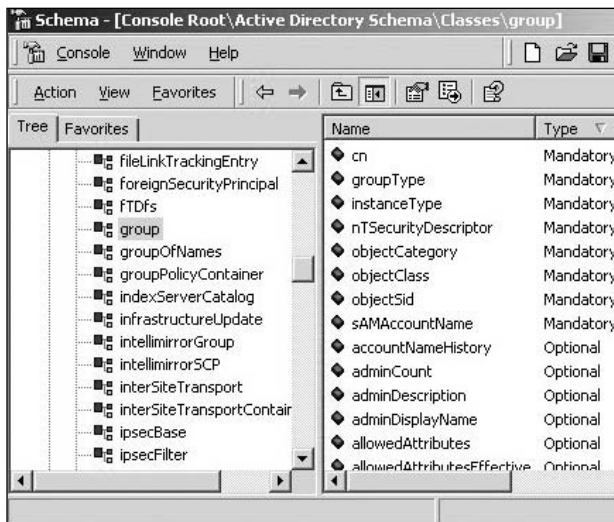



Figura 7: Lista degli attributi della classe group

5.10 LE PROPERTY CACHE INTERFACES

Nel capitolo precedente abbiamo mostrato il funzionamento, a grandi linee, della Property Cache.
Si è detto che ogni qualvolta effettuiamo il binding ad un oggetto di AD, tutti i suoi attributi sono scaricati nella cache locale per utilizzi futuri. ADSI mette a disposizione le interfacce IADsPropertyList, IADsPropertyEntry, IADsPropertyValue e IADsPropertyValue2 che ci con-

sentono di recuperare molte informazioni sugli attributi di un oggetto/classe.
In particolare:

- IADsPropertyList: consente di leggere e modificare una lista di proprietà contenute all'interno della Property Cache;
- IADsPropertyEntry: utilizzata per la modifica di una specifica entry nella Property Cache;
- IADsPropertyValue: consente di scrivere e leggere il valore di una proprietà esprimendolo nel formato previsto da un determinato data type;
- IADsPropertyValue2: simile alla precedente, ma in grado di sfruttare ed esprimere una proprietà in un qualunque tipo di data type.

Innanzitutto diamo un'occhiata ai metodi specifici e di nostro interesse di ognuna:

IADsPropertyList (proprietà/metodi)	**Descrizione**
get_PropertyCount	Ritorna il numero di proprietà dell'oggetto.
Next	Passa all'item/proprietà successivo.
Skip	Salta un determinato numero di item nella lista delle proprietà.
Reset	Ritorna all'inizio della lista.
Item	Ritorna ad una proprietà specificata attraverso l'indice o il nome.
GetPropertyItem	Ritorna il valore di una proprietà.

IADsPropertyList (proprietà/metodi)	Descrizione
PutPropertyItem	Modifica il valore di una proprietà.
ResetPropertyItem	Resetta I valori di una determinate proprietà.
PurgePropertyList	Elimina tutte le proprietà dalla lista.

Tabella 7: Proprietà e metodi dell'interfaccia IADsPropertyList

IADsPropertyEntry (proprietà)	Descrizione
get/put_Name	Preleva/imposta il nome di una proprietà.
get/put_ADS_Type	Preleva/imposta il valore che esprime il tipo di dato di una proprietà.
get/put_ControlCode	Flag che identifica o imposta eventuali modifiche sull'entry.
get/put_Values	Preleva/imposta i valori correnti di una determinata entry.

Tabella 8: Proprietà dell'interfaccia IADsPropertyEntry

IADsPropertyValue (proprietà/metodi)	Descrizione
Clear0	Cancella il valore corrente.
get/put_ADsType	Preleva/imposta il tipo di dato.
get/put_DNString	Preleva/imposta il DN dell'oggetto.
get/put_CaseExactString	Preleva/imposta il valore di una stringa case-sensitive.
get/put_CaseIgnoreString	Preleva/imposta il valore di una stringa non case-insensitive.

Tabella 9: Metodi e proprietà dell'interfaccia IADsPropertyValue

IADsPropertyValue (proprietà/metodi)	Descrizione
get/put_PrintableString	Preleva/imposta il valore di una printable string.
get/put_NumericString	Preleva/imposta il valore di una stringa numerica.
get/put_Boolean	Preleva/imposta il valore di un boolean.
get/put_Integer	Preleva/imposta il valore di un intero.
get/put_OctetString	Preleva/imposta il valore di un ottetto (8 bit).
get/put_SecurityDescriptor	Preleva/imposta il valore di un security descriptor.
get/put_LargeInteger	Preleva/imposta il valore di un intero espresso con 64 bit.
get/put_UTCTime	Preleva/imposta il valore di un Coordinated Universal Time.

Tabella 10: Metodi e proprietà dell'interfaccia IADsPropertyValue

IADsPropertyValue2 (metodi)	Descrizione
GetObjectProperty	Recupera i valori di una determinata proprietà.
PutObjectProperty	Imposta i valori di una determinate proprietà.

Tabella 11: Metodi dell'interfaccia IADsPropertyValue2

In Tabella 7 e Tabella 9, in rosso, sono indicati rispettivamente l'unica proprietà e l'unico metodo dell'interfaccia alla quale la tabella si riferisce. Affinché si abbia un'idea ancor più precisa di cosa vogliano effettivamente indicare ciascuno degli oggetti ADSI appena mostrati, riportiamo di seguito una piccola schematizzazione che può aiutare a capire meglio questi aspetti:

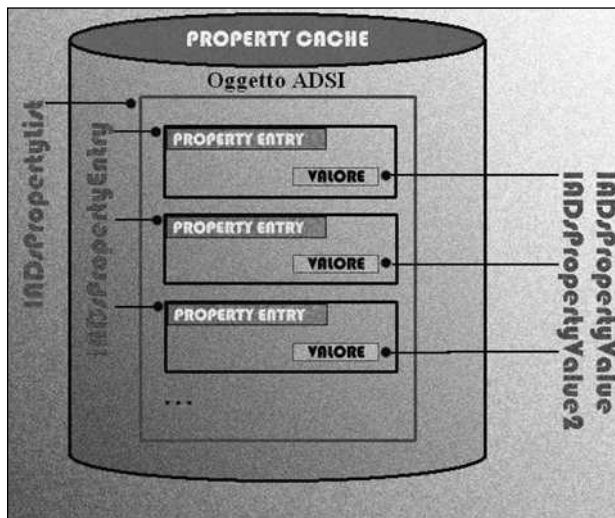



Figura 8: Property Cache Interfaces

Tanto per avere un'idea ancor più precisa sull'utilizzo di queste interfacce, proviamo a mostrare un semplice esempio dal quale poi faremo le dovute considerazioni:

```
Option Explicit
Dim objGrp
Dim Cont
Dim pCont
Dim objPropEntry
Dim objPropValue
Dim arrADSType(29)
arrADSType(0)  = "ADSTYPE_INVALID"
arrADSType(1)  = "ADSTYPE_DN_STRING"
arrADSType(2)  = "ADSTYPE_CASE_EXACT_STRING"
arrADSType(3)  =  "ADSTYPE_CASE_IGNORE_STRING"
arrADSType(4)  =  "ADSTYPE_PRINTABLE_STRING"
```

```
arrADSType(5)  =  "ADSTYPE_NUMERIC_STRING"
arrADSType(6)  =  "ADSTYPE_BOOLEAN"
arrADSType(7)  =  "ADSTYPE_INTEGER"
arrADSType(8)  =  "ADSTYPE_OCTET_STRING"
arrADSType(9)  =  "ADSTYPE_UTC_TIME"
arrADSType(10) =  "ADSTYPE_LARGE_INTEGER"
arrADSType(11) =  "ADSTYPE_PROV_SPECIFIC"
arrADSType(12) =  "ADSTYPE_OBJECT_CLASS"
arrADSType(13) =  "ADSTYPE_CASEIGNORE_LIST"
arrADSType(14) =  "ADSTYPE_OCTET_LIST"
arrADSType(15) =  "ADSTYPE_PATH"
arrADSType(16) =  "ADSTYPE_POSTALADDRESS"
arrADSType(17) =  "ADSTYPE_TIMESTAMP"
arrADSType(18) =  "ADSTYPE_BACKLINK"
arrADSType(19) =  "ADSTYPE_TYPEDNAME"
arrADSType(20) =  "ADSTYPE_HOLD"
arrADSType(21) =  "ADSTYPE_NETADDRESS"
arrADSType(22) =  "ADSTYPE_REPLICAPOINTER"
arrADSType(23) =  "ADSTYPE_FAXNUMBER"
arrADSType(24) =  "ADSTYPE_EMAIL"
arrADSType(25) =  "ADSTYPE_NT_SECURITY_DESCRIPTOR"
arrADSType(26) =  "ADSTYPE_UNKNOWN"
arrADSType(27) =  "ADSTYPE_DN_WITH_BINARY"
arrADSType(28) = "ADSTYPE_DN_WITH_STRING"
```

' Binding al gruppo Administrators del dominio MyDomain.it.

```
Set  obj Grp=
GetObject("LDAP://CN=Administrators,CN=Builtin,DC=MyDomain,DC=it")
objGrp.GetInfo
WScript.Echo  ">
ELENCO PROPRIETA' DEL GRUPPO ADMINISTRATORS <"
& vbCrLf
```

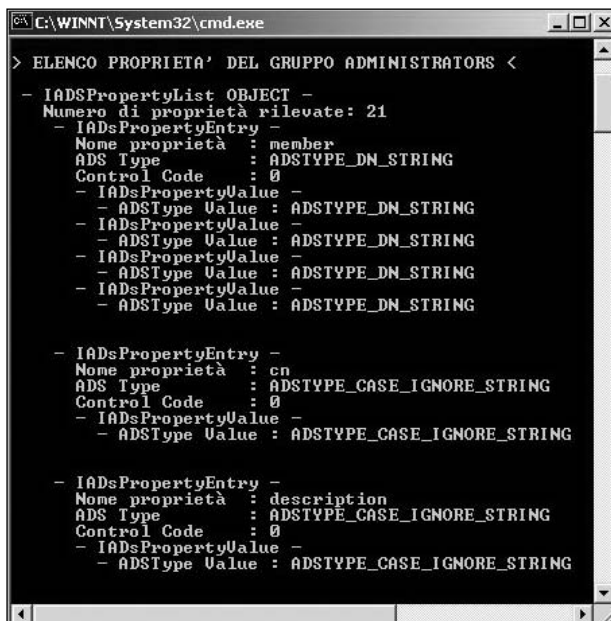
Per tutte le proprietà rilevate, fa vedere i valori attuali.
' NB: Alcune proprietà sono, in realtà, degli array.
Questo rende necessario il secondo ciclo FOR
NFORMAZIONI GENERALI (INTERFACCIA IADsPropertyList)

```
Wscript.Echo " - IADSPropertyList OBJECT - "
Wscript.Echo "   Numero di proprietà rilevate: " & objGrp.PropertyCount
```

Elenca alcune informazioni per ogni proprietà rilevata

```
For Cont = 0 To objGrp.PropertyCount - 1
   pCont = 1
```

INFORMAZIONI GENERALI (INTERFACCIA IADsPropertyEntry)

```
      Set objPropEntry = objGrp.Item(Cont)
      Wscript.Echo "     - IADsPropertyEntry -"
      Wscript.Echo "       Nome proprietà  : " & objPropEntry.Name
      Wscript.Echo "       ADS Type        : " &
arrADSType(objPropEntry.ADSType)
      Wscript.Echo "       Control Code    : "
& objPropEntry.ControlCode
For Each objPropValue In objPropEntry.Values
  Wscript.Echo "       - IADsPropertyValue -"
Wscript.Echo "         - ADSType Value : " &
arrADSType(objPropValue.ADSType)
   Next
   Wscript.Echo vbCrlf
Next
```

LISTATO 9 Esempio d'utilizzo delle Property Cache Interfaces

Ecco l'output parziale prodotto dallo script:

```
C:\WINNT\System32\cmd.exe

> ELENCO PROPRIETA' DEL GRUPPO ADMINISTRATORS <

 - IADSPropertyList OBJECT -
   Numero di proprietà rilevate: 21
    - IADsPropertyEntry -
      Nome proprietà   : member
      ADS Type         : ADSTYPE_DN_STRING
      Control Code     : 0
      - IADsPropertyValue -
        - ADSType Value : ADSTYPE_DN_STRING
      - IADsPropertyValue -
        - ADSType Value : ADSTYPE_DN_STRING
      - IADsPropertyValue -
        - ADSType Value : ADSTYPE_DN_STRING
      - IADsPropertyValue -
        - ADSType Value : ADSTYPE_DN_STRING


    - IADsPropertyEntry -
      Nome proprietà   : cn
      ADS Type         : ADSTYPE_CASE_IGNORE_STRING
      Control Code     : 0
      - IADsPropertyValue -
        - ADSType Value : ADSTYPE_CASE_IGNORE_STRING


    - IADsPropertyEntry -
      Nome proprietà   : description
      ADS Type         : ADSTYPE_CASE_IGNORE_STRING
      Control Code     : 0
      - IADsPropertyValue -
        - ADSType Value : ADSTYPE_CASE_IGNORE_STRING
```

Figura 9: Output dello script

Come si può vedere, accanto alla stringa Nome proprietà, sono elencati gli stessi nomi degli attributi già visti all'inizio del capitolo, ma ottenuti non utilizzando lo Schema di Active Directory, ma interrogando direttamente la Property Cache. Tuttavia, anche in considerazione di quest'ultima affermazione e dando un'occhiata più attenta al risultato del listato precedente, possiamo desumere ed aggiungere alcuni particolari importanti:

- se confrontiamo l'output dei due script precedenti, ci renderemo presto conto che il numero di attributi ritornati dal secondo è un sottoinsieme del primo. Questo risultato ci porta ad affermare una cosa importante: gli attributi che non contengono alcun valore non vengono scaricati in cache. In que-

sti casi, il codice di errore riportato all'interno dello script è rappresentato dalla costante Const E_ADS_PROPERTY_NOT_FOUND = &H8000500D.

- l'array arrADSType è l'array che definisce le costanti che identificano il possibile valore della proprietà ADSType degli oggetti ADSI che li supportano. In particolare, l'indice del generico item dell'array indica effettivamente il valore assunto dalla generica costante che definisce il tipo di dato. Ad esempio, se prendiamo la riga arrADSType(7) = "ADSTYPE_INTEGER" questa ci dice che il valore della costante ADSTYPE_INTEGER è proprio 7. Stesso discorso, ovviamente, per i restanti. Queste costanti, in realtà, fanno parte di una struttura di tipo Enum denominata ADSTYPEENUM e dalla quale, per comodità, sono state estrapolate le costanti.
- Il ciclo:

```
For Cont = 0 To objGrp.PropertyCount – 1
…
Next
```

all'interno del quale vengono rilevate tutte le proprietà della cache, poteva anche essere sostituito con qualcosa di simile a:

```
Set objPropEntry = objGrp.Next
While
(Not (IsNull(objPropEntry)) And Err.Number = 0)
        …
Set objPropEntry = objGrp.Next
Wend
```

La prima riga richiama il metodo Next dell'oggetto objGrp che, trattandosi della prima volta che il metodo viene chiamato, assegnerà a objPropEntry una copia del primo elemento della lista di proprietà.

- Se osserviamo bene l'output dello script mostrato (ma anche se abbiamo dato un'attenta occhiata a metodi e proprietà delle interfacce oggetto di questo capitolo) ci saremo accorti che la proprietà ADSType è presente sia per la IADsPropertyEntry che per la IADsPropertyValue. Tuttavia, rivedendo l'output parziale dello script mostrato in Figura 9, non è difficile rilevare che il risultato, in corrispondenza di questa proprietà, è sempre lo stesso, identico per entrambe le interfacce. Questo non rappresenta un vero e proprio errore, bensì l'implementazione di una feature che Active Directory è predisposta ad "accogliere", ma non ancora supportata: la possibilità, per il generico item "IADsPropertyValue" di gestire collezioni di valori di diverso tipo.
- L'oggetto IADsPropertyEntry possiede una proprietà denominata ControlCode che identifica in che maniera va trattata quella proprietà quando verrà effettuato l'aggiornamento in Active Directory. Il valore che può assumere questo flag è identificato da 4 costanti:

  - ADS_PROPERTY_CLEAR = 1
  - ADS_PROPERTY_UPDATE = 2
  - ADS_PROPERTY_APPEND = 3
  - ADS_PROPERTY_DELETE = 4

che, se ricordiamo bene, sono proprio quelle che utilizzavamo con il metodo PutEx dell'interfaccia IADs e definite dalla struttura di tipo Enum denominata ADS_PROPERTY_OPERATION_ENUM.

- I valori rappresentati dalla proprietà ADSType, in realtà, mappano in una qualche maniera altrettanti valori LDAP presenti in Active Directory. Semplificando il discorso, questo è spiegabile perché Active Directory possiede un certo numero di data type particolari, difficilmente rappresentabili nel

modo COM-ADSI e, di conseguenza, questo significa occorre creare una certa corrispondenza (laddove possibile) tra i tipi di dati definiti/definibili da un oggetto ADSI e quelli riconosciuti da LDAP/Active Directory. Di seguito, per inciso, la tabella delle corrispondenze:

ADSTYPE	LDAP Type
ADSTYPE_DN_STRING	LDAPTYPE_DN
ADSTYPE_CASE_EXACT_STRING	LDAPTYPE_CASEEXACTSTRING
ADSTYPE_CASE_IGNORE_STRING	LDAPTYPE_CASEIGNORESTRING LDAPTYPE_BITSTRING LDAPTYPE_DIRECTORYSTRING LDAPTYPE_COUNTRYSTRING LDAPTYPE_IA5STRING LDAPTYPE_OID DAPTYPE_TELEPHONENUMBER LDAPTYPE_ATTRIBUTETYPEDESCRIPTION LDAPTYPE_OBJECTCLASSDESCRIPTION
ADSTYPE_CASE_IGNORE_STRING	LDAPTYPE_POSTALADDRESS LDAPTYPE_DELIVERYMETHOD LDAPTYPE_ENHANCEDGUIDE LDAPTYPE_FACSIMILETELEPHONENUMBER LDAPTYPE_GUIDE LDAPTYPE_NAMEANDOPTIONALUID LDAPTYPE_PRESENTATIONADDRESS LDAPTYPE_TELEXNUMBER LDAPTYPE_DSAQUALITYSYNTAX LDAPTYPE_DATAQUALITYSYNTAX LDAPTYPE_MAILPREFERENCE LDAPTYPE_OTHERMAILBOX LDAPTYPE_ACCESSPOINTDN LDAPTYPE_ORNAME
ADSTYPE_PRINTABLE_STRING	LDAPTYPE_PRINTABLESTRING
ADSTYPE_NUMERIC_STRING	LDAPTYPE_NUMERICSTRING
ADSTYPE_BOOLEAN	LDAPTYPE_BOOLEAN

Tabella 12: Corrispondenza tipi ADSI-LDAP

ADSTYPE	LDAP Type
ADSTYPE_INTEGER	LDAPTYPE_INTEGER
ADSTYPE_OCTET_STRING	LDAPTYPE_OCTETSTRING LDAPTYPE_CERTIFICATE LDAPTYPE_CERTIFICATELIST LDAPTYPE_CERTIFICATEPAIR LDAPTYPE_PASSWORD LDAPTYPE_OID LDAPTYPE_TELETEXTERMINALIDENTIFIER LDAPTYPE_AUDIO LDAPTYPE_JPEG LDAPTYPE_FAX
ADSTYPE_NT_SECURITY_ _DESCRIPTOR	LDAPTYPE_SECURITY_DESCRIPTOR
ADSTYPE_UTC_TIME	LDAPTPYE_UTCTIME
ADSTYPE_UTC_TIME	LDAPTYPE_GENERALIZEDTIME
ADSTYPE_LARGE_INTEGER	LDAPTYPE_INTEGER8
E_ADS_CANT_CONVERT_ _DATATYPE	LDATYPE_UNKNOWN

Tabella 12: Corrispondenza tipi ADSI-LDAP

ACTIVE DIRECTORY SEARCHING

Sinora abbiamo sempre dato per scontato il fatto di conoscere esattamente l'oggetto sul quale operare. Non ci siamo mai posti il problema di dover ricercare qualcosa all'interno di Active Directory che rispondesse a determinati criteri e sul quale, successivamente, compiere le nostre azioni. La maniera più comoda e quella sicuramente consigliata per chi deve implementare ricerche all'interno dei propri script VBS è quella di utilizzare l'ADSI OLE DB Provider e su questa tecnologia incentreremo questo capitolo.

Alcune premesse

Sinora, quando abbiamo avuto la necessità di agire su un determi-

nato oggetto di Active Directory, abbiamo anche dovuto conoscere quell'oggetto sul quale lavorare. Nel capitolo 7 abbiamo visto come sia possibile accedere agli item di un determinato container e, addirittura, filtrare gli elementi in base a determinati criteri, ma questo meccanismo, sebbene perfettamente funzionante, si può rivelare molto lento e comunque non sempre adatto ai nostri scopi.
Nei casi in cui abbiamo necessità di recuperare informazioni solo su oggetti che rispondono a determinate caratteristiche attraverso script VBS, dobbiamo affidarci ad un'ulteriore strumento a disposizione con ADSI e denominato ADSI OLE DB provider. In questo capitolo vedremo alcune tecniche che ci consentono di utilizzare questi oggetti per effettuare ricerche. Ovviamente, mostreremo solo alcuni dei metodi utili a raggiungere lo scopo, rimandando per maggiori informazioni al sito della Microsoft.

5.11 RICERCHE CON ADO

ADO rappresenta una tecnologia attraverso la quale è possibile ricercare informazioni all'interno di un database, recuperando tali dati attraverso un'apposito oggetto definito genericamente Resultset.
Per facilitare l'integrazione con Active Directory, in particolare, e sfruttare le capacità offerte da ADO, la Microsoft ha costruito un ADO Database Provider per ADSI, definito prima con il nome di ADSI OLE DB Provider. Purtroppo, cominciamo con il dire subito che l'ADSI OLE DB Provider consente operazioni in sola lettura, ma a discapito di questo "difetto", offre possibilità estremamente superiori alle tecniche viste in precedenza con altri oggetti come l'IADsContainer.
Il modello ad oggetti che descrive ADO è composto da 9 elementi, tra i quali menzioniamo gli oggetti Connection, Command e Recordset tra i più importanti che incontreremo.
Quando decidiamo di affidarci ad ADO per le ricerche all'interno di Active Directory, possiamo/dobbiamo rispettare almeno quattro passi:

- Connessione al database di AD.
- Definizione ed avvio della query.
- Lettura ed utilizzo dei risultati.
- Chiusura della connessione

Per quanto riguarda il primo passo, vediamo immediatamente una piccola porzione di codice che ci consente di comprendere meglio come attuare questa fase (tralasciamo per ora la dichiarazione delle variabili in testa allo script):

```
Utente = "CN=Administrator,CN=Users,dc=MyDomain,dc=it"
Password = ""
Set objConn = CreateObject("ADODB.Connection")
objConn.Provider = "ADSDSOObject"
' ------------------------------------------------------
objConn.Open "",Utente, Password
' ------------------------------------------------------
'Controlla l'esito dell'autenticazione
If objConn.State = adStateOpen Then
  WScript.Echo "Connection OK!" & vbCrlf
Else
  WScript.Echo "Connection NOT OK!"
  WScript.Quit(1)
End If
```

Come si può facilmente intuire, le istruzioni prima della sezione tratteggiata, non fanno altro che dichiarare il nome utente e la password da utilizzare nella connessione che sfrutteremo successivamente attraverso il metodo Open() dell’oggetto ADODB.
Connection. Inutile sottolineare anche che l’oggetto ADSDSOObject è l’oggetto che prima avevamo chiamato ADSI OLE DB Provider.
A questo punto, se tutto è andato bene, abbiamo finalmente stabilito la connessione al DB di Active Directory.

Non ci resta che interrogarlo, con i metodi opportuni.

5.12 QUERY VERSO AD

Adesso arriva certamente la parte più interessante di questo capitolo ossia il modo attraverso il quale formulare una query qualunque. Partiamo innanzitutto da quello che vogliamo ottenere. Immaginiamo di voler elencare tutti gli elementi del dominio ossia ottenere la seguente lista:

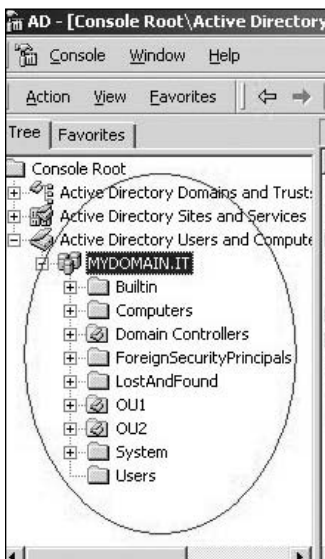



Figura 10: I container dell'oggetto MyDomain.it

Per ottenere questa lista, la nostra query deve equivalere a qualcosa del tipo: "cercami tutti gli oggetti che si trovano all'interno del contenitore MYDOMAIN.IT, senza elencare il contenuto di ciascuno e mostrami nome e stringa ADSPath".

Ecco la stringa equivalente a quanto richiesto, ma scritta secondo

la sintassi riconosciuta da ADO/ADSI:

```
"<LDAP://dc=MyDomain,dc=it>;;Name,ADSPath;Onelevel"
```

La stringa che identifica la query precedente,come si può "vedere", è composta da quattro parti, separate ciascuna dal carattere ';':

- Base della ricerca: identifica il punto di partenza da dove verrà avviata la ricerca. Nel nostro caso si parte dalla radice del dominio d'esempio Mydomain.it;
- Criteri filtro: qui vengono impostati i criteri per filtrare opportunamente i valori da cercare. Nel caso appena visto, non sono stati impostati criteri (ossia la query ritorna tutto quello che trova);
- Attributi: specifica gli attributi che devono essere inclusi all'interno della tabella dei risultati (nel nostro caso, solo Name e ADSPath);
- Scope: questo parametro (opzionale) è molto importante perché specifica il raggio d'azione della query. I suoi possibili valori possono essere:

  - Base
  - OneLevel
  - SubTree

Nel caso non venga specificato nulla, il valore di default è assunto come SubTree.
Di seguito un disegno (figura 11) che mostra in maniera chiara la differenza tra un scope e l'altro. Volendo riportare quanto detto sottoforma d'istruzione, ecco cosa va aggiunto allo script precedente

```
Set objRS = objConn.Execute (
"<LDAP://dc=MyDomain,dc=it>;;Name,ADSPath;Onelevel")
```

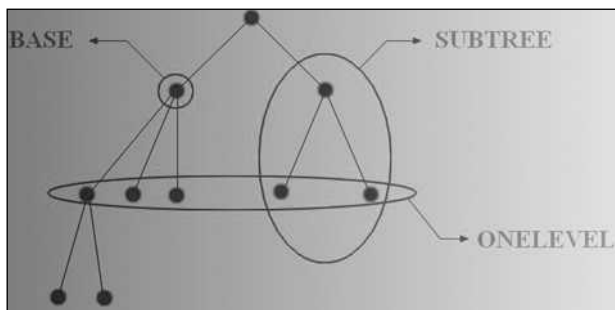
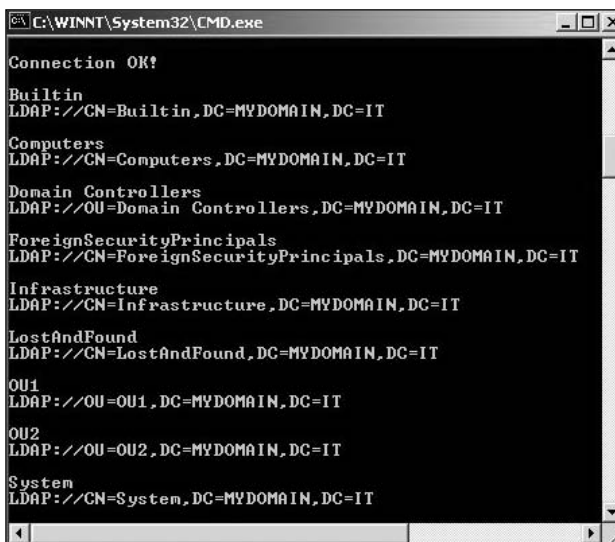



Figura 11: Scope di una query

A questo punto l'oggetto objConn ci ritorna una tabella (recordset) di risultati che possiamo consultare come se ti trattasse di una tabella formata dalle due colonne Name e ADSPath.

5.13 CONSULTARE I RISULTATI

Chi ha esperienza di programmazione in Visual Basic o, comunque, ha già avuto a che fare con ADO, non faticherà moto a comprendere i metodi principali per muoversi all'interno di un recordset. Ecco, in definitiva, l'ultima parte dello script, comprendente anche la chiusura della connessione:

```
While Not objRS.EOF
    Wscript.Echo objRS.Fields.Item("Name").
Value
    Wscript.Echo objRS.Fields.Item("ADSPath").
Value & vbCrlf
    objRS.MoveNext
Wend
 objConn.Close
```

che produce un risultato parziale simile a questo:

Figura 12: Output dello script

Adesso abbiamo un'idea di come affrontare il problema e possiamo andare un pò più nel dettaglio.

5.14 ADO LDAP DIALECT

La query utilizzata nello script precedente, abbiamo visto essere scritta in una forma che "ricorda" molto la sintassi LDAP. In realtà, quando, dobbiamo costruire la stringa che identifica i nostri criteri di ricerca mediante ADO, con ADSI possiamo utilizzare due tipi di sintassi, meglio noti con il nome di dialetti (Dialect):

- LDAP Dialect.
- SQL Dialect.

Ognuno di questi dialetti prevede una forma (sintassi) utile a poter scrivere la query che consentirà di recuperare le informazioni da Ac-

tive Directory. In questa sede ci concentreremo sulla prima tipologia, l'LDAP Dialect perché la sua sintassi è meno intuibile rispetto a quella utilizzata dal dialetto SQL che probabilmente ognuno di noi conosce. Non ripetiamo quanto detto in precedenza circa la forma della generica stringa consente di ritornare il recordset con i dati della query, ma ci concentrermo direttamente sul secondo parametro di quella stringa, quello che avevamo denominato Criteri filtro.
Volendolo rappresentare secondo una forma generalizzata, potremmo scrivere così la stringa che permette di definire questo secondo parametro:

```
<Filtro 1> <Filtro 2> … <Filtro N>
```

Il generico Filtro X è una stringa di questo tipo:

```
(<operatore>(Criterio 1) (Criterio 2) … (Criterio 3))
```

Malgrado la notazione utilizzata possa essere considerata approssimativa, dovrebbe comunque rendere l'idea sulla forma "strana" di questa sintassi. Innanzitutto vediamo l'operatore, che va posto all'inizio e non tra gli elementi della query. Gli operatori booleani previsti sono tre:

```
& AND
| OR
! NOT
```

Questo significa che se vogliamo cercare oggetti che rispondono, ad esempio, sia a Criterio 1 che a Criterio 2 (AND), dovremo scrivere qualcosa del tipo:

```
(& (Criterio1) (Criterio 2))
```

Stesso discorso se invece avessimo voluto cercare oggetti che ri-

spondono o al Criterio 1 oppure al Criterio 2 (OR):

```
(| (Criterio 1) (Criterio 2))
```

Infine, se volessimo cercare oggetti, ad esempio, che rispondono a Criterio 1 e rispondono o al Criterio 2 o al Criterio 3, scriveremmo:

```
(&(Criterio 1) (|(Criterio 2) (Criterio 3)))
```

Ora che abbiamo compreso come posizionare ciascun elemento "legato" al generico operatore e facente parte della query, dobbiamo vedere a cosa equivale realmente l'oggetto definito genericamente Criterio X. Esso può essere suddiviso in tre parti specifiche:

```
<Attributo> <Operatore di comparazione> <Valore>
```

dove:

- Attributo: nome dell'attributo degli oggetti di Active Directory da considerare;
- Operatore di comparazione: Active Directory riconosce soltanto tre operatori ossia =, >= e <= che non credo abbiano bisogno di spiegazioni;
- Valore: ovviamente rappresenta il valore dell'attributo che deve essere utilizzato come criterio di ricerca. Per poter costruire la corretta stringa che identifica il valore da cercare, è possibile utilizzare anche l'operatore '*' che ci consente di rintracciare valori che iniziano, finiscono o contengono una certa stringa. Ad esempio:

- cn=Fra* Tutti gli oggetti che posseggono l'attributo cn che inizia per Fra;
- cn=*Fra* Tutti gli oggetti che posseggono l'attributo

cn che contien Fra;

- cn=*Fra Tutti gli oggetti che posseggono l'attributo cn che finiscono per Fra;

Oltre a questo, possiamo utilizzare anche altri caratteri speciali ed in particolare il carattere '\' che, può essere seguito dal valore esadecimale (espresso con 2 cifre) di un carattere qualunque, consente di rappresentarlo.
Ad esempio "\28ABC\29" equivale a "(ABC)".
Adesso, alla luce di quanto appena detto sinora, mostriamo lo script precedente, completo anche nel secondo parametro.
Per ipotesi, stiamo supponendo di voler ricercare tutti gli oggetti marcati come container del dominio (ricordiamo che il test lo si fa controllando l'attributo objectClass e vedendo se equivale a 'container' o 'organizationalUnit'.

```
Option Explicit
Const adStateOpen = 1
Dim objConn
Dim objRS
Dim Utente
Dim Password
Dim Object
```

Imposta l'utente e la password per la connessione

```
Utente = "CN=Administrator,CN=Users,dc=MyDomain,dc=it"
Password = ""
```

Crea l'oggetto ADO Connection ed apri la connessione con utente e password

```
Set objConn = CreateObject("ADODB.Connection")
```

```
objConn.Provider = "ADSDSOObject"
objConn.Open "",Utente, Password
```

Controlla l'esito dell'autenticazione

```
If objConn.State = adStateOpen Then
  WScript.Echo "Connection OK!" & vbCrlf
Else
  WScript.Echo "Connection NOT OK!"
  WScript.Quit(1)
End If
```

Elenca tutti i container e le Organizational Unit

```
Set objRS = objConn.Execute ("<LDAP://dc=MyDomain,dc=it>; _
(|(objectClass=container)(objectClass=organizationalunit));
Name,ADSPath;OneLevel")
```

Scorri gli elementi del Recordset sino alla fine

```
While Not objRS.EOF
  Wscript.Echo objRS.Fields.Item("Name").Value
  Wscript.Echo objRS.Fields.Item("ADSPath").Value & vbCrlf
  objRS.MoveNext
Wend
objConn.Close
```

LISTATO 10 Script per ottenere l'elenco di Container e OU

Considerando lo script appena esposto, aggiungiamo ancora qualche piccolo dettaglio:

- Se per distrazione inserissimo uno spazio tra objectClass e

'=', il risultato sarebbe diverso dal previsto, come se quell'uguaglianza non esistesse. Pertanto occorre prestare attenzione a questo genere di errori perché spesso, casi come questo, fanno perdere molto tempo.

- Nell'esempio proposto abbiamo considerato una querry scritta secondo il dialetto LDAP. A titolo di curiosità mostriamo l'equivalente stringa scritta in linguaggio SQL like e sicuramente meno ostica della precedente:

```
'Set objRS = objConn.Execute
("SELECT Name, ADsPath FROM
'LDAP://dc=MyDomain,dc=it' WHERE ObjectClass ='Container' OR
ObjectClass='OrganizationalUnit'").
```

- Nello script ci accontentiamo semplicemente di leggere i risultati riportati all'interno del recordset, senza modificarne alcuno. Ecco una semplice istruzione che, inserita all'interno del ciclo FOR…NEXT che scorre il recordset permette di "agganciare" ciascun item e modificarlo con i metodi visti nei precedenti capitoli:

```
Set Object = GetObject(objRS.Fields.Item("ADSPath").Value)
```

5.15 L'OGGETTO COMMAND

I metodi visti in precedenza per recuperare informazioni da Active Directory con ADO non sono gli unici o, perlomeno, non sono tutti uguali a quello visto.
All'inizio abbiamo accennato circa l'esistenza, all'interno dell'Object Model di ADO, di un oggetto denominato Command.
Attraverso i metodi e le proprietà di questo oggetto, possiamo passare ad una connessione già attiva, diversi comandi e molto di più.
A titolo d'esempio, mostriamo brevemente la prima parte dello script

precedente all'interno del quale abbiamo modificato il dialetto utilizzato e mostriamo l'utilizzo del metodo Open dell'oggetto Recordset:

```
Set objRS = CreateObject("ADODB.Recordset")
strQuery = "SELECT Name, ADsPath FROM '
LDAP://dc=MyDomain,dc=it' WHERE ObjectClass ='Container'
OR ObjectClass='OrganizationalUnit'"

 objRS.ActiveConnection = objConn
```

' Elenca tutti i container e le Organizational Unit

```
objRS.Open strQuery, objConn
```

Se lanciamo lo script precedente a quello appena mostrato, con le modifiche che abbiamo appena visto, noteremo che i risultati mostrati sono l'elenco di tutti gli oggetti ritrovati a partire dalla Search Base specificata. In realtà, esiste un metodo per specificare ulteriori parametri alla nostra query utilizzando l'oggetto Command di ADO, incluso il numero di elementi che deve ritornare.
L'oggetto Command possiede diverse proprietà. Tuttavia, quelle che di solito vengono utilizzate più di frequente sono:

- Sort on: in Active Directory consente di definire un attributo che servirà all'ordinamento dei risultati;
- Size limit: valore intero che definisce il numero di elementi da ritornare;
- SearchScope: identifica il tipo di ricerca da effettuare (Base, OneLevel o SubTree) ed è rappresentato dalle seguenti costanti:
  - ADS_SCOPE_BASE = 0
  - ADS_SCOPE_ONELEVEL = 1

- ADS_SCOPE_SUBTREE = 2

- Timeout: specifica il tempo (espresso in secondi) che il client deve attendere prima di decidere di terminare la ricerca

A questo punto, mostriamo la porzione di codice equivalente al precedente, ma modificata utilizzando l'oggetto Command:

```
Set objConn = CreateObject("ADODB.Connection")
objConn.Provider = "ADSDSOObject"
objConn.Open "",Utente, Password
```

Crea l'oggetto Command e "legalo" all'oggetto Connection precedente

```
Set objCommand = CreateObject("ADODB.Command")
Set objCommand.ActiveConnection = objConn
```

Imposta la stringa della query e, di conseguenza, la proprietà CommandText

```
strQuery = "SELECT Name, ADsPath FROM 'LDAP:
//dc=MyDomain,dc=it'" &_

 "WHERE ObjectClass ='Container' OR ObjectClass='OrganizationalUnit'"
objCommand.CommandText = strQuery
```

Elenca tutti i container e le Organizational Unit

```
Set objRS = objCommand.Execute()
```

Da quello che si può dedurre, quindi, il risultato è pressocchè equivalente. Vediamo, in definitiva, lo script completo e modificato in-

troducendo anche i nuovi “controlli” sui risultati.

```
Option Explicit

Const adStateOpen = 1

Const ADS_SCOPE_BASE = 0
Const ADS_SCOPE_ONELEVEL = 1
Const ADS_SCOPE_SUBTREE = 2
```

Imposta a 10 gli elementi da ritornare

```
Const ItemReturned = 10
Dim objConn
Dim objRS
Dim Utente
Dim Password
Dim Object
Dim strQuery
Dim objCommand
```

Imposta l’utente e la password per la connessione

```
Utente = "CN=Administrator,
CN=Users,dc=MyDomain,dc=it"
Password = ""
```

Crea l’oggetto ADO Connection ed apri la connessione con utente e password

```
Set objConn = CreateObject("ADODB.Connection")
objConn.Provider = "ADSDSOObject"
objConn.Open "",Utente, Password
```

Controlla l'esito dell'operazione di connessione

```
Controlla l'esito dell'autenticazione
If objConn.State = adStateOpen Then
  WScript.Echo "Connection OK!" & vbCrlf
Else
  WScript.Echo "Connection NOT OK!"
  WScript.Quit(1)
End If
```

Crea l'oggetto Command e "legalo" all'oggetto Connection precedente

```
Set objCommand = CreateObject("ADODB.Command")
Set objCommand.ActiveConnection = objConn
```

Imposta le proprietà dell'oggetto Command

```
strQuery =
 "SELECT Name, ADsPath FROM 'LDAP:
//dc=MyDomain,dc=it'" &_
    "WHERE ObjectClass ='Container'
OR ObjectClass='OrganizationalUnit'"
objCommand.CommandText = strQuery
objCommand.Properties("Sort on") = "Name"
objCommand.Properties("Size Limit") = ItemReturned
objCommand.Properties("SearchScope") = ADS_SCOPE_SUBTREE
```

Elenca tutti i container e le Organizational Unit

```
Set objRS = objCommand.Execute()
```

Scorri gli elementi del Recordset sino alla fine

```
While Not objRS.EOF
    Wscript.Echo objRS.Fields.Item("Name").Value
    Wscript.Echo objRS.Fields.Item("ADSPath").Value & vbCrlf
    Set Object = GetObject(objRS.Fields.Item("ADSPath").Value)
```

FA QUALCOSA CON L'OGGETTO, POI PASSA AL SUCCESSIVO…

```
  objRS.MoveNext
Wend
```

Chiudi la connessione e "distruggi" gli oggetti creati

```
objConn.Close
Set objCommand = Nothing
Set objRS = Nothing
Set Object = Nothing
```

LISTATO 11 Script di ricerca con ADO

5.16 ALCUNI CONSIGLI

All'inizio di questo capitolo, quando abbiamo visto come definire una generica query su Active Directory, abbiamo detto che la generica stringa che indicava i parametri della ricerca, era composta al massimo da quattro parametri.
In realtà, esiste un ulteriore parametro che possiamo indicare al suo interno migliorando la gestione delle ricerche qualora, tra i parametri richiesti, ci siano anche attributi multivalore.
Il nome di questo parametro è Range Limit e, se specificato, occupa la penultima posizione all'interno della stringa che definisce la query.
In definitiva, essa assumerebbe la seguente forma:

```
Search Base; Search Filter; Attributes List; Range Limit; Scope
```

La forma generale di un Range Limit è la seguente:

```
<Nome Attributo>;Range=range
```

Facciamo a questo proposito un esempio generico che ci consenta di capire meglio quanto appena affermato e facciamolo considerando l'attributo member che elenca i nomi dei componenti di un certo gruppo, limitando ai primi 100 gli elementi ritornati:

```
"<LDAP://cn=Users,dc=MyDomain,dc=com>;
(objectCategory=group);member;Range=0-99;Base"
```

Ecco,in definitiva, il listato completo di uno script che elenca i primi 10 membri di ogni gruppo trovato a partire da dalla Search Base cn=Users,dc=MyDomain,dc=it..

```
Option Explicit
Const adStateOpen = 1
Const ADS_SCOPE_BASE = 0
Const ADS_SCOPE_ONELEVEL = 1
Const ADS_SCOPE_SUBTREE = 2
Const ItemReturned = 100

Dim objConn
 Dim objRS
Dim Utente
Dim Password
Dim ListaMembri
Dim strQuery
Dim objCommand
Dim Membro
```

Imposta l'utente e la password per la connessione

```
Utente = "CN=Administrator,CN=Users,dc=MyDomain,
dc=it"
Password = ""
Set objConn = CreateObject("ADODB.Connection")
objConn.Provider = "ADSDSOObject"
objConn.Open "",Utente, Password
```

Controlla l'esito dell'operazione di connessione

```
If objConn.State = adStateOpen Then
  WScript.Echo "Connection OK!" & vbCrlf
Else
  WScript.Echo "Connection NOT OK!"
  WScript.Quit(1)
End If
```

Crea l'oggetto Command e "legalo" all'oggetto Connection precedente

```
Set objCommand = CreateObject("ADODB.Command")
Set objCommand.ActiveConnection = objConn
```

Imposta le proprietà dell'oggetto Command

```
strQuery =
"<LDAP://cn=Users,dc=MyDomain,dc=it>;
(ObjectCategory=group);
Name,
member; Range=0-9"
objCommand.CommandText = strQuery
objCommand.Properties("Sort on") = "Name"
objCommand.Properties("Size Limit") = ItemReturned
objCommand.Properties("SearchScope") = ADS_SCOPE_SUBTREE
```

Elenca tutti membri di ogni gruppo trovato

```
Set objRS = objCommand.Execute()
```

Scorri gli elementi del Recordset sino alla fine

```
 While Not objRS.EOF
    Wscript.Echo objRS.Fields.Item("Name").Value
```

ATTENZIONE: Il campo con indice 2 non è member, ma equivale esattamente alla stringa

```
member;Range=0-9
ossia l'intera stringa Range Limit
ListaMembri = objRS.Fields(1).Value
```

Se non ci sono membri…

```
    On Error Resume Next
    For each Membro In ListaMembri
        Wscript.Echo " ---->" & Membro
    Next
    objRS.MoveNext
Wend
```

Chiudi la connessione e "distruggi" gli oggetti creati

```
objConn.Close
Set objConn = Nothing
Set objRS = Nothing
Set objCommand = Nothing
```

LISTATO 12 Dimostrazione dell'uso del parametro Range Limit

Per poter utilizzare il Range Limit con esito positivo, occorre tener presente, però, che tutti quegli attributi che posseggono un numero di elementi inferiore al numero massimo stabilito, non vengono considerati. Mostriamo un esempio forse più chiaro di tante parole. Supponiamo di avere solo 4 gruppi denominati Gruppo1, Gruppo2, Gruppo3 e Gruppo4, ciascuno contenente un numero di elementi pari rispettivamente a 1, 2, 3 e 4.
Indichiamo nella prima colonna della tabella che segue i range stabiliti (supponendo di averli modificati di volta in volta nello script precedente) e vediamone i risultati ossia quanti elementi lo script mostrerà a video:0

Gruppo	N.Membri	0-1	0-2	0-3
Gruppo 1	1	#	#	#
Gruppo 2	2	#	#	#
Gruppo 3	3	2	#	#
Gruppo 4	4	2	3	#

Tabella 13: Corrispondenze tra Range Limit e risultato visualizzato

Quello che viene mostrato, dunque, è certamente il numero di elementi indicati dal massimo valore del range + 1, ma prendendo in considerazione soltanto i gruppi che contengono membri in numero superiore.
Un ulteriore accorgimento che occorre tener presente riguarda le performance legate ad una generica query.
Quando avviamo una ricerca all'interno di Active Directory, quello che va ricordato sempre è che trattasi di una query avviata in un ambiente "ricco" di oggetti e, se la stringa che la definisce non è impostata correttamente, occorrerà attendere parecchio per vederne i risultati.
A questo proposito avevamo visto anche come pilotare qualche parametro che poteva influenzare le nostre ricerche, attraverso le proprietà dell'oggetto Command, ma l'accorgimento migliore è chia-

ramente all'interno della stringa che definisce i criteri di ricerca. Ecco alcuni semplici e noti consigli a tal proposito:

- Utilizzare un solo attributo indicizzato per query. L'esempio classico è objectCategory (del quale parleremo fra poco). Per poter vedere se un attributo è indicizzato oppure no, possiamo utilizzare MMC come mostrato di seguito.

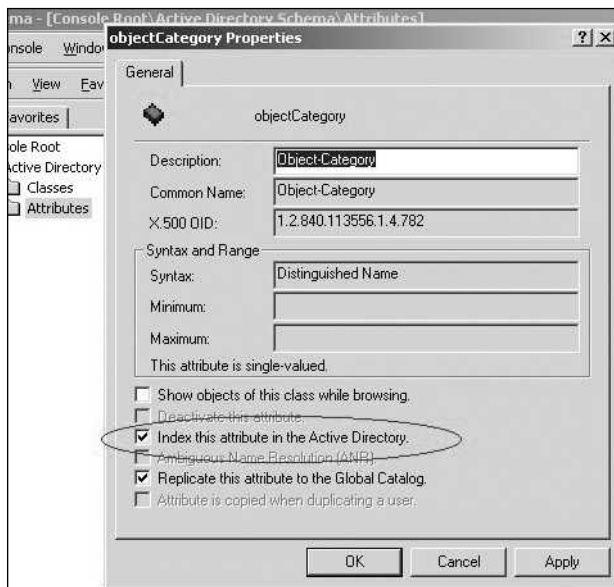



Figura 13: Un tipico errore non gestito dall'applicazione.

- Cercare di utilizzare sempre combinazioni di objectCategory ed objectClass nelle proprie query. Il primo è a singolo valore ed indica proprio la tipologia dell'oggetto (persona, contatto, computer, ecc.). Il secondo è multivalore e contiene tutte le classi a cui appartiene quell'oggetto (si tenga a mente il fatto che una generica classe all'interno dello schema

è la risultante di altre classi dalle quali è stata derivata). Pertanto la combinazione di uesti due attributi può esserci utile a restringere maggiormente il campo d'azione della query.

- Laddove possibile limitare la stringa relativa al confronto. Ad esempio, se c'interessano tutti i gruppi che iniziano con Grp è inutile fare qualcosa del tipo (Name= GrpTestLocal1 OR Name= GrpTestLocal2, ecc.). Utilizzare forme simili a Name=Grp*.
- Impostare sempre i giusti parametri relativi allo scope della ricerca e al limite massimo di elementi da recuperare che ci interessano realmente.
- L'oggetto Recordset possiede anch'esso una proprietà Filter (chi ha già utilizzato ADO lo saprà certamente) che può tornarci utile per filtrare i risultati

Un ultimo piccolo consiglio: quando definiamo un oggetto Connection all'interno di uno script, a meno che non sia strettamente necessario, sfruttiamo sempre lo stesso anche per ricerche successive e distruggiamolo solo al termine.
Stesso discorso per l'oggetto Command. Questo piccolo accorgimento può aiutarci a gestire meglio le risorse aumentando anche i tempi di risposta del nostro programma.

ESEMPI PRATICI

Siamo finalmente giunti alla terza ed ultima parte di questo libro. Non è certamente un mistero il fatto che su tantissimi argomenti era necessario più spazio o che tanti altri sono stati, volutamente, tralasciati per dar spazio ad argomenti diversi.
Tuttavia chiunque si avvicini a questo nuovo mondo avrà certamente incontrato qualche difficoltà in meno nella comprensione dei capitoli precedenti, perché si è tentato di privilegiare la chiarezza a scapito della quantità di argomenti trattati. I capitoli che sinora sono stati letti dovrebbero aver reso l'idea di quanto giri attorno al servizio di directory di Windows 2000 ed al mondo della programmazione mediante ADSI e dovrebbero aver fornito a tutti le basi e gli strumenti necessari per poter lavorare con essi.
Naturalmente, come qualunque argomento di questa complessità, va approfondito ulteriormente leggendo la documentazione inerente e provando, facendo test.
Ed è proprio per questi motivi che in quest'ultima parte abbiamo voluto dar spazio all'aspetto prettamente pratico, raccogliendo diversi script che potranno essere d'esempio e d'utilità ad ognuno di voi. Per ciascun listato, laddove si è ritenuto necessario, sono state aggiunte delle annotazioni/spiegazioni a compendio di quanto già descritto attraverso il codice ed i commenti in essi contenuti.

Listato 1. – Elencare i Namespace supportati

```
Option Explicit

Dim objADS
Dim objADSChild
Set objADS = GetObject("ADS:")
```

Per ogni namespace trovato…

```
For Each objADSChild in objADS
WScript.Echo objADSChild.Name

Next
```

Distruggi gli oggetti

```
Set objADS = Nothing
Set objADSChild = Nothing
```

Listato 2. – Elencare le proprietà dell'oggetto RootDSE

```
Option Explicit
Dim objDSE
Dim Cont
Dim pCont
DIm strPropName
Dim objPropEntry
Dim varPropValue
Dim objPropValue
```

Binding all'oggetto RootDSE

```
Set objDSE = GetObject("LDAP://rootDSE")
objDSE.GetInfo
WScript.Echo "> ELENCO PROPRIETA'
[" & objDSE.PropertyCount & "]
DI ROOTDSE. <" & vbCrLf
```

Per tutte le proprietà rilevate, fa vedere i valori attuali.
NB: Alcune proprietà sono, in realtà, degli array.
Questo rende necessario il secondo ciclo FOR.

```
For Cont = 0 To objDSE.PropertyCount - 1
    pCont=1
    Set objPropEntry = objDSE.Item(Cont)
    strPropName = objPropEntry.Name
    WScript.Echo "--------------------------------------"
    WScript.Echo "- "& strPropName
    WScript.Echo "--------------------------------------"

    For Each objPropValue In objPropEntry.Values

        WScript.Echo pCont & ") " &
 objPropValue.GetObjectProperty(objPropEntry.ADsType)
       strPropName = vbNull
        pCont = pCont+1

    Next

    Wscript.Echo vbCrLf
Next
```

Distruggi gli oggetti

```
Set objDSE = Nothing
Set objPropEntry = Nothing
Set objPropValue = Nothing
```

Listato 3. – Identificare la modalità di dominio

```
Option Explicit
Dim objDSE
Dim strLDAP
Dim objDomain
```

Binding al dominio

```
Set objDSE = GetObject("LDAP://rootDSE")
strLDAP =
"LDAP://" & objDSE.Get("defaultNamingContext")
Set objDomain = GetObject(strLDAP)
```

Controlla se il dominio è in modalità MIXED o NATIVE

```
If objDomain.Get("nTMixedDomain") > 0 Then
Wscript.Echo "La modalità del dominio è MIXED."
```

Togliendo il commento alle due istruzioni successive, possiamo passare dalla modalità MIXED a quella NATIVE (irreversibile).

```
   objDomain.Put "nTMixedDomain", 0
   ' objDomain.SetInfo
Else
  Wscript.Echo "La modalità del dominio è NATIVE"
 End if
```

Distruggi gli oggetti

```
Set objDSE = Nothing
Set objDomain = Nothing
```

Listato 4. – Visualizzare lo stato del Global Catalog

```
Option Explicit
Dim objRootDSE
Set objRootDSE= GetObject("LDAP://RootDSE")
Wscript.Echo "Stato GC: " & objRootDSE.Get("isGlobalCatalogReady")
```

Distruggi gli oggetti

```
Set objRootDSE = Nothing
```

Listato 5. – Creare un nuovo utente

Questo script crea un nuovo utente senza metterlo in alcun gruppo. Per poter avviare lo script correttamente, lanciare CSCRIPT <Nome Script> <Nome utente da creare> <Password>.

```
Option Explicit
Dim objDSE
Dim strDefaultDN
Dim objContainer
Dim strName
Dim strPass
Dim objUser
```

Controlla se sono stati inseriti i parametri

```
If Wscript.Arguments.count <> 2 Then
  Wscript.Echo "Il numero di parametri inseriti è errato"
  Wscript.Echo "Lanciare lo script con i parametri
<NomeUtente>
<Password>"
  Wscript.Quit
End If
```

Memorizza i parametri passati per utilizzarli in seguito

```
strName = Wscript.arguments(0)
strPass = Wscript.arguments(1)
```

Effettua il binding alla root del dominio

```
Set objDSE = GetObject("LDAP://rootDSE")
strDefaultDN =
objDSE.Get("defaultNamingContext")
msgbox strDefaultDN
Set objContainer = GetObject("LDAP://" & strDefaultDN)
```

Crea innanzitutto l'utente…

```
Set objUser = objContainer.Create("user", "CN=" & strName)
```

imposta alcuni parametri (i primi 2 sono essenziali)

```
objUser.Put "sAMAccountName", strName
objUser.Put "userAccountControl", &H200
objUser.Put "description",
 "Questo è un nuovo utente"
```

Salva i dati e successivamente imposta la password

```
objUser.SetInfo
```

NB: L'impostazione della pwd avviene solo DOPO aver effettuato una commit sui precedenti dati

' Imposta la password

```
objUser.SetPassword (strPass)
```

Distruggi gli oggetti

```
Set objUser = Nothing
Set objContainer = Nothing
Set objDSE = Nothing
```

Listato 6.– Disabilitare un utente

Questo script disabilita un utente che per ipotesi si trova sotto il contenitore Users.

```
Option Explicit
Dim objUsr
Dim strName
Dim FlagUAC
Dim Arg
Const ADS_UF_ACCOUNTDISABLE = 2
Const ABILITA = 1
Const DISABILITA = 0
```

Controlla se sono stati inseriti i parametri

```
If Wscript.Arguments.count <> 2 Then
   Wscript.Echo "Il numero di parametri inseriti è errato"
   Wscript.Echo "Lanciare lo script con i parametri <NomeUtente> <0/1>"
   Wscript.Echo "0 = DISABILITA"
   Wscript.Echo "1 = ABILITA"
   Wscript.Quit
End If
```

Memorizza il nome dell'utente e l'operazione da effettuare

```
strName = Wscript.arguments(0)
Arg = Wscript.Arguments(1)
```

Crea l'oggetto relativo all'utente da disabilitare

```
Set objUsr = GetObject("LDAP://CN=
" & strName & ",CN=Users,DC=MyDomain,DC=it")
FlagUAC = objUsr.Get("userAccountControl")
```

Imposta l'operazione da effettuare in base al secondo parametro

```
Select Case int(Arg)
        Case ABILITA
```

1) Metodo...

```
  objUsr.Put "userAccountControl", FlagUAC AND ABILITA
```

2) Metodo..Togliere i commenti e metterli all'istruzione precedente per usare questo metodo (Interfaccia IADsUser)

```
objUsr.AccountDisabled = FALSE
Case DISABILITA
```

1) Metodo

```
objUsr.Put "userAccountControl",
FlagUAC OR ADS_UF_ACCOUNTDISABLE
```

2) Metodo..Togliere i commenti e metterli all'istruzione precedente per usare questo metodo (Interfaccia IADsUser)

```
objUsr.AccountDisabled = TRUE
End Select
```

Memorizza le nuove impostazioni

```
objUsr.SetInfo
```

Distruggi l'oggetto objUsr

```
Set objUsr = Nothing
```

Listato 7.– Unlock di un utente

```
Option Explicit
Dim objUsr
Dim FlagUAC
```

La costante che permette d'impostare il bit relativo al lock dell'utente è Const ADS_UF_LOCKOUT = &H0010, che in binario equivale a 10000.
Per sbloccare l'utente, quindi, è necessario effettuare un AND logico tra userAccountControl ed il valore &H15, equivalente al valore binario 01111.

```
Const UNLOCK = &H15
```

Modificare la stringa di binding in base all'utente

```
Set objUsr =
GetObject("LDAP://CN=UtenteLocked,CN=Users,
DC=MyDomain,DC=it")
FlagUAC = objUsr.Get("userAccountControl")
```

1) Metodo...

```
objUsr.Put "userAccountControl", FlagUAC AND UNLOCK
```

2) Metodo..Togliere i commenti e metterli all'istruzione precedente per usare questo metodo

```
objUsr.IsAccountLocked = FALSE
objUsr.SetInfo
```

Distruggi l'oggetto

```
Set objUsr = Nothing
```

Listato 8.– Elencare gli utenti disabilitati

In questo listato vengono elencati gli utenti disabilitati sfruttando il binding “solito” che sfrutta il security context dell’utente che sta avviando lo script (correntemente autenticato al dominio). Nel listato successivo è mostrata la prima parte di questo script opportunamente modificata per mostrare come utilizzare il metodo OpenDSObject, specificando credenziali diverse.

```
Option Explicit
Const ADS_UF_ACCOUNTDISABLE = 2
Const ADS_SCOPE_BASE = 0
Const ADS_SCOPE_ONELEVEL = 1
Const ADS_SCOPE_SUBTREE = 2
Const PageSize = 100
Const Timeout = 30
Const CacheRes = False
Dim objRootDSE
Dim objConnection
Dim objCommand
Dim objRS
Dim strQuery
Dim FlagUAC
Dim NroAccountLocked
NroAccountLocked = 0
Binding al dominio
Set objRootDSE = GetObject("LDAP://rootDSE")
```

Creazione dell'oggetto Connection per la query con ADO

```
Set objConnection = CreateObject("ADODB.Connection")
objConnection.Open  "Provider=ADsDSOObject;"
```

Creazione dell'oggetto Command

```
Set objCommand = CreateObject("ADODB.Command")
```

Imposta i parametri utili all'oggetto

```
strQuery = "SELECT distinguishedName, userAccountControl,
Name FROM 'LDAP://" &
  objRootDSE.Get("defaultNamingContext")
& "'" & " WHERE objectCategory='User'"
objCommand.CommandText = strQuery
objCommand.ActiveConnection = objConnection
objCommand.Properties("Page Size") = PageSize
objCommand.Properties("SearchScope") = ADS_SCOPE_SUBTREE
objCommand.Properties("Timeout") = Timeout
objCommand.Properties("Cache Results") = CacheRes
```

Avvia la query

```
Set objRS = objCommand.Execute
```

Leggi i risultati

```
 Do Until objRS.EOF
FlagUAC = objRS.Fields("userAccountControl")
If FlagUAC AND ADS_UF_ACCOUNTDISABLE Then
NroAccountLocked = NroAccountLocked + 1
 WScript.Echo NroAccountLocked & ")
 " & objRS.Fields("Name") & " è disabilitato"
   End If
   objRS.MoveNext
Loop
WScript.Echo VbCrLf &
```

```
"Il totale degli account rilevati è " & NroAccountLocked
```

Chiudi la connessione

```
objConnection.Close
```

Distruggi gli oggetti

```
Set objRootDSE = Nothing
Set objConnection = Nothing
Set objCommand = Nothing
Set objRS = Nothing
```

Listato 9.– Elencare gli utenti disabilitati (con autenticazione)

```
Option Explicit
Const ADS_UF_ACCOUNTDISABLE = 2
Const ADS_SCOPE_BASE = 0
Const ADS_SCOPE_ONELEVEL = 1
Const ADS_SCOPE_SUBTREE = 2
Const PageSize = 100
Const Timeout = 30
Const CacheRes = False
Dim objRootDSE
Dim objConnection
Dim objCommand
Dim Utente
Dim Password
Dim objRS
Dim strQuery
Dim FlagUAC
Dim NroAccountLocked
```

Costanti utili per OpenDSObject

```
Const ADS_SECURE_AUTHENTICATION = &H1
Const ADS_USE_ENCRYPTION = &H2
Const ADS_USE_SSL = &H2
Const ADS_READONLY_SERVER = &H4
Const ADS_PROMPT_CREDENTIALS = &H8
Const ADS_NO_AUTHENTICATION = &H10
Const ADS_FAST_BIND = &H20
Const ADS_USE_SIGNING = &H40
Const ADS_USE_SEALING = &H80
Const ADS_USE_DELEGATION = &H100
Const ADS_SERVER_BIND = &H200
NroAccountLocked = 0
```

Specifica il nome dell'utente e la password in particolare: il nome utente può essere specificato in diversi modi.

Distinguished Name:
cn=UtenteGenerico,cn=Users,dc=Mydomain,dc=it
User Account: UtenteGenerico
User Principal Name (UPN): UtenteGenerico@MyDomain.it
Domain\User: MyDomain\UtenteGenerico

```
Utente = "cn=UtenteGenerico,cn=Users,
dc=Mydomain,dc=it"
Password = " "
```

Binding al dominio con autenticazione

```
Set objRootDSE = GetObject("LDAP:")
Set objRootDSE = objRootDSE.OpenDSObject
("LDAP://RootDSE",Utente,Password,ADS_SECURE_AUTHENTICATION)
```

Listato 10.– Elencare i gruppi di appartenenza di un utente

Questo script mostra l'elenco dei gruppi di appartenenza di un utente che si trova sotto il contenitore Users.

```
Dim objRootDSE
Dim SingleGroup
Dim strDefaultDN
Dim strUser
Dim objContainer
```

Controlla se è stato passato il nome dell'utente

```
  If Wscript.Arguments.count = 0 Then
     strUser = InputBox
 ("Inserisci il nome dell'utente da controllare:")
     If strUser = "" Then WScript.Quit(1)
  Else
     strUser = Wscript.Arguments(0)
  End If
```

Binding all'oggetto

```
Set objRootDSE = GetObject("LDAP://rootDSE")
strDefaultDN = objRootDSE.Get("defaultNamingContext")
Set objContainer = GetObject("LDAP://CN=" & strUser & ",
CN=Users," & strDefaultDN)
```

Richiesta informazioni

```
objContainer.GetInfo
```

Output dei gruppi di appartenenza

```
Wscript.Echo "Elenco dei gruppi di appartenenza"
For Each SingleGroup in objContainer.Groups
Wscript.Echo SingleGroup.Name
    Next
```

Distruggi gli oggetti

```
Set objRootDSE = Nothing
Set objContainer = Nothing
```

Listato 11.– Eliminare i gruppi di un utente

```
Option Explicit
On Error Resume Next
Dim objUser
Dim objGroup
Dim arrMemberOf
Dim Group
Const ADS_PROPERTY_CLEAR = 1
Const ADS_PROPERTY_DELETE = 4
Const E_ADS_PROPERTY_NOT_FOUND  = &H8000500D
```

Elimina i gruppi ai quali appartiene l'utente Utente1

```
Set objUser = GetObject("LDAP://cn=Utente1,
cn=Users,dc=My    Domain,dc=it")
arrMemberOf = objUser.GetEx("memberOf")
```

Se la proprietà non è valorizzata nella Property Cache

```
If Err.Number = E_ADS_PROPERTY_NOT_FOUND Then
  WScript.Echo "Nessun elemento trovato"
```

```
  WScript.Quit
End If
```

Per ciascun gruppo trovato, effettua il binding ed elimina l'item che corrisponde all'utente Utente1 (fatta eccezione per il Primary Group)

```
For Each Group In arrMemberOf
 Set objGroup = GetObject("LDAP://" & Group)
objGroup.PutEx ADS_PROPERTY_DELETE, "member",
Array("cn=Utente1,cn=Users,dc=MyDomain,dc=it")
     objGroup.SetInfo
Next
```

Distruggi gli oggetti

```
Set objUser = Nothing
Set objGroup = Nothing
```

Listato 12.– Elencare tutti gli utenti che iniziano con un certo prefisso

```
Option Explicit
Dim objRootDSE
Dim strDomainDN
Dim objConnection
Dim objCommand
Dim strQuery
Dim objRS
Dim Cont
Dim Prefix
Const ADS_SCOPE_BASE = 0
Const ADS_SCOPE_ONELEVEL = 1
Const ADS_SCOPE_SUBTREE = 2
```

```
Const PageSize = 100
Const Timeout = 30
Const CacheRes = False
```

Criterio che il CN dell'utente deve soddisfare

```
Prefix = "Francesco*"
```

Binding all'oggetto RootDSE

```
Set objRootDSE = GetObject("LDAP://rootDSE")
strDomainDN = objRootDSE.Get("defaultNamingContext")
```

Crea l'oggetto Connection

```
Set objConnection = CreateObject("ADODB.Connection")
objConnection.Provider = "ADsDSOObject"
objConnection.Open
```

Crea l'oggetto Command

```
Set objCommand = CreateObject("ADODB.Command")
Set objCommand.ActiveConnection = objConnection
```

Imposta la query

```
strQuery = "SELECT cn FROM 'LDAP://" & strDomainDN &
"' WHERE objectCategory='user' AND cn='" & Prefix & "'"
```

Imposta i parametri dell'oggetto Command

```
objCommand.CommandText = strQuery
objCommand.Properties("Page Size") = PageSize
```

```
objCommand.Properties("SearchScope") = ADS_SCOPE_SUBTREE
objCommand.Properties("Timeout") = Timeout
objCommand.Properties("Cache Results") = CacheRes
    Set objRS = objCommand.Execute
Cont = 0
If objRS.EOF Then
  WScript.Echo "Nessun utente risponde ai criteri impostati"
Else
 While Not objRS.EOF
Cont = Cont + 1
     WScript.Echo Cont & ": " & objRS.Fields("cn")
            objRS.MoveNext
 Wend
End If
```

Chiudi la connessione

objRS.Close
objConnection.Close

Distruggi gli oggetti

```
Set objRootDSE = Nothing
Set objConnection = Nothing
Set objCommand = Nothing
Set objRS = Nothing
```

Listato 13.– Creare un gruppo e inserire un utente

```
Option Explicit
Dim objDSE
Dim strDefaultDN
```

```
Dim objContainer
Dim objGroup
```

Effettua il binding al dominio

```
Set objDSE = GetObject("LDAP://rootDSE")
strDefaultDN = objDSE.Get("defaultNamingContext")
```

Effettua il binding a Users

```
Set objDSE = GetObject("LDAP://rootDSE")
Set objContainer = GetObject
("LDAP://cn=Users," & strDefaultDN)
Set objGroup = objContainer.Create
("Group", "cn=GruppoDiTest")
```

Impostazione degli attributi necessari per la creazione del gruppo

```
objGroup.Put "sAMAccountName","GruppoDiTest"
objGroup.SetInfo
strDefaultDN = ",cn=Users," & strDefaultDN
```

Inserimento dell'utente Guest

```
objGroup.Put "member",
"cn=Guest" & strDefaultDN
  objGroup.SetInfo
```

Distruggi gli oggetti

```
Set objDSE = Nothing
Set objContainer = Nothing
Set objGroup = Nothing
```

Listato 14.– Creare un gruppo e assegnargli uno Scope

```
Option Explicit

Dim objRootDSE
Dim objGroup
Dim strDomain
Dim objContainer

Dim GrpTypeFlag1, GrpTypeFlag2

Identifica i Security group, ' distinguendoli dai Distribution group
Const ADS_GROUP_TYPE_SECURITY_ENABLED = &H8000000
Identifica i vari tipi di gruppi che si possono creare
Const ADS_GROUP_TYPE_GLOBAL_GROUP = &H2
Global group
Const ADS_GROUP_TYPE_LOCAL_GROUP = &H4
Local group
Const ADS_GROUP_TYPE_UNIVERSAL_GROUP = &H8                '
Universal group

Imposta il flag che definisce il tipo di gruppo (Security o Distribution)
' 0 = Distribution
' 1 = Security
GrpTypeFlag1 = 0
Imposta il flag che definisce il tipo di gruppo (global, local o universal)
GrpTypeFlag2 = ADS_GROUP_TYPE_GLOBAL_GROUP
```

Effettua il binding al dominio

```
Set objRootDSE = GetObject("LDAP://RootDSE")
strDomain = objRootDSE.Get("defaultNamingContext")
```

```
Set objContainer = GetObject("LDAP://cn=Users," & strDomain)
```

Crea il gruppo ed assegnagli gli attributi obbligatori

```
Set objGroup = objContainer.Create
("Group","cn=MyFirstGroup")
objGroup.Put "sAMAccountName",
 "MyFirstGroup"
```

Imposta il tipo di gruppo (scope)

```
If GrpTypeFlag1 = 0 Then
```

Distribution Group

```
   objGroup.Put "groupType", GrpTypeFlag2
Else
```

Security Group

```
      objGroup.Put "groupType", GrpTypeFlag2 Or
ADS_GROUP_TYPE_SECURITY_ENABLED
End If
```

Aggiorna le informazioni

```
objGroup.SetInfo
```

Distruggi gli oggetti

```
Set objRootDSE = Nothing
Set objGroup = Nothing
Set objContainer = Nothing
```

Listato 15.– Aggiungere altri gruppi ad un gruppo

Questo script aggiunge i gruppi GrpTest1 e GrpTest2 al gruppo GrpLocalTest.

```
Option Explicit
Const ADS_PROPERTY_APPEND = 3
Dim objRootDSE
Dim objGroup
Dim strConfigNameContext
```

Effettua il binding al gruppo GrpLocalTest

```
Set objRootDSE = GetObject("LDAP://RootDSE")
strConfigNameContext = objRootDSE.
Get("defaultNamingContext")
Set objGroup =
GetObject("LDAP://cn=GrpLocalTest,cn=Users,"
& strConfigNameContext)
```

Aggiungi i due gruppi GrpTest1 e GrpTest2

```
objGroup.PutEx ADS_PROPERTY_APPEND, "member",
 Array("cn=GrpTest1,cn=Users,dc=MyDomain,
dc=it", _
        "cn=GrpTest2,cn=Users,dc=MyDomain,
dc=it")
objGroup.SetInfo
```

Distruggi gli oggetti

```
Set objRootDSE = NOthing
Set objGroup = Nothing
```

Listato 16.– Eliminare alcuni o tutti i membri da un gruppo

```
Option Explicit
Const ADS_PROPERTY_CLEAR = 1
Const ADS_PROPERTY_DELETE = 4
Dim objRootDSE
Dim objGroup
Dim strConfigNameContext
```

Binding al gruppo GrpLocalTest

```
Set objRootDSE =
GetObject("LDAP://RootDSE")
strConfigNameContext =
objRootDSE.Get("defaultNamingContext")
 Set objGroup =
GetObject("LDAP://cn=GrpLocalTest,cn=Users,"
& strConfigNameContext)
```

Eliminazione degli item GrpTest1 e GrpTest2. Per eliminare TUTTI gli elementi di un gruppo, utilizzare l'istruzione seguente: objGroup.PutEx ADS_PROPERTY_CLEAR, "member", 0

```
objGroup.PutEx ADS_PROPERTY_DELETE, "member",
Array

("cn=GrpTest1,cn=Users,dc=MyDomain,dc=it", _
        "cn=GrpTest2,cn=Users,dc=MyDomain,dc=it")

objGroup.SetInfo
Set objRootDSE = Nothing
Set objGroup = Nothing
```

Listato 17.– Elencare le proprietà del gruppo Administrators

```
Option Explicit
Dim objDSE
Dim Cont
Dim pCont
Dim strPropName
Dim objPropEntry
Dim objPropValue
Const    ADSTYPE_OCTET_STRING = 8
Const    ADSTYPE_LARGE_INTEGER = 10
Const    ADSTYPE_NT_SECURITY_DESCRIPTOR = 25
```

Binding al gruppo Administrators

```
Set objDSE =
GetObject("LDAP://CN=Administrators,CN=Builtin,
DC=MyDomain,DC=it")
objDSE.GetInfo
WScript.Echo "> ELENCO PROPRIETA' DEL GRUPPO ADMINISTRATORS <"
& vbCrLf
```

Per tutte le proprietà rilevate, fa vedere i valori attuali.
NB: Alcune proprietà sono, in realtà, degli array. Questo rende necessario il secondo ciclo FOR

```
For Cont = 0 To objDSE.PropertyCount - 1
    pCont=1
    Set objPropEntry = objDSE.Item(Cont)
    strPropName = objPropEntry.Name
    WScript.Echo "--------------------------------------"
    WScript.Echo "- "& strPropName
```

```
    WScript.Echo "-----------------------------------------"
    ' Per ogni proprietà, rileva il tipo di dato
    For Each objPropValue In objPropEntry.Values
      Select Case objPropValue.ADsType
         Case ADSTYPE_OCTET_STRING
          WScript.Echo pCont & ") " & "<OCTET STRING>"
             Case ADSTYPE_LARGE_INTEGER
                  WScript.Echo pCont & ") " & "<LARGE INTEGER>"
               Case ADSTYPE_NT_SECURITY_DESCRIPTOR
          WScript.Echo pCont & ") " &
  "<Security Descriptor>"
       Case Else
     WScript.Echo pCont & ") " &
   objPropValue.GetObjectProperty(objPropEntry.ADsType)
       End Select
          strPropName = vbNull
        pCont = pCont+1

    Next

    Wscript.Echo vbCrLf
Next
```

Distruggi gli oggetti

```
Set objDSE = Nothing
Set objPropEntry = Nothing
Set objPropValue = Nothing
```

Listato 18.– Elencare le proprietà IAD di utenti e gruppi

Questo script si prefigge come scopo di elencare a video le proprietà esposte dall’interfaccia IADs di tutti i gruppi principali di Windows 2000.

Lo script memorizza in un array questi nomi e li sfrutta per effettuare il binding ad essi. Il codice mostrato, così com'è, non è ovviamente molto flessibile e poteva essere scritto diversamente ed in maniera migliore.
Tuttavia, si è pensato che in questo modo, possa risultare più comprensibile a chi ha poca confidenza con ADSI.

```
Option Explicit
Dim objRootDSE, objIAD
Dim BuiltInUG(17)
Dim Cont
Dim strLDAP
ReturnIADsProp
 Private Sub ReturnIADsProp
```

Memorizza in un array il nome dei principali gruppi di Windows 2000.

```
BuiltInUG(0) ="Administrator"
BuiltInUG(1) ="Cert Publishers"
BuiltInUG(2) ="DnsAdmins"
BuiltInUG(3) ="DnsUpdateProxy"
BuiltInUG(4) ="Domain Admins"
BuiltInUG(5) ="Domain Computers"
BuiltInUG(6) ="Domain Controllers"
BuiltInUG(7) ="Domain Guests"
BuiltInUG(8) ="Domain Users"
BuiltInUG(9) ="Enterprise Admins"
BuiltInUG(10)="Group Policy Creator Owners"
BuiltInUG(11)="Guest"
BuiltInUG(12)="IUSR_SERVER"
BuiltInUG(13)="IWAM_SERVER"
BuiltInUG(14)="krbtgt"
```

```
BuiltInUG(15)="RAS and IAS Servers"
BuiltInUG(16)="TsInternetUser"
```

Effettua il binding alla root.

```
Set objRootDSE = GetObject("LDAP://rootDSE")
```

Per ogni gruppo, visualizza le informazioni dell'interfaccia IADs

```
For Cont = 0 To 16
```

Effettua il binding al gruppo

```
 strLDAP = "LDAP://CN=
  " & BuiltInUG(Cont) & ",CN=Users," & _
      objRootDSE.Get("defaultNamingContext")
        Set objIAD = GetObject(strLDAP)
   Wscript.Echo "------------------------------------"
     Wscript.Echo "- " & BuiltInUG(Cont)
     Wscript.Echo "------------------------------------"
     WScript.Echo "AdsPath:"  & objIAD.AdsPath
     WScript.Echo "Class  :"  & objIAD.Class
     WScript.Echo "GUID   :"  & objIAD.GUID
     WScript.Echo "Name   :"  & objIAD.Name
     WScript.Echo "Parent :"  & objIAD.Parent
     WScript.Echo "Schema :"  & objIAD.Schema & vbCrlf
    Next
End Sub
```

Distruggi gli oggetti

```
Set objRootDSE = Nothing
Set objIAD = Nothing
```

Listato 19.– Elencare tutte le OU del dominio (senza ADO)

```
Option Explicit
Dim objRootDSE
Dim strLDAP
Dim Object
Dim objDomain
Dim sClass
Dim Recursive
Effettua il binding al dominio
Set objDSE = GetObject("LDAP://rootDSE")
strLDAP = "LDAP://" & objDSE.Get("defaultNamingContext")
Set objDomain = GetObject(strLDAP)
Call ListObject(objDomain)
Sub ListObject(objContainer)
Per ogni container trovato, verifica se trattasi di OU
  For Each Object In objContainer
    For Each sClass in Object.ObjectClass
    If sClass = "organizationalUnit" Then
Wscript.Echo "Object Name : "
  & vbTab & Object.Name & vbCrLf & _
  "Object ADSPath : " &  vbTab & Object.AdsPath & vbCrLf & _
    "Object Parent : " &  vbTab & Object.Parent
       ListObject(Object)
          End If
      Next
  Next
End Sub
' Distruggi gli oggetti
Set objRootDSE = Nothing
Set objDomain = Nothing
Set Object = Nothing
```

Listato 20.– Elencare tutte le OU del dominio (con ADO)

```
Option Explicit
Const adStateOpen = 1
Const ADS_SCOPE_BASE = 0
Const ADS_SCOPE_ONELEVEL = 1
Const ADS_SCOPE_SUBTREE = 2
Const ItemReturned = 10
Dim objConn
Dim objRS
Dim strDomain
Dim objRootDSE
Dim strQuery
Dim objCommand
Binding all'oggetto RootDSE
Set objRootDSE = GetObject("LDAP://rootDSE")
Recupero del nome di dominio
strDomain = "LDAP://" & objRootDSE.Get("defaultNamingContext")
```

Creazionde dell'oggetto Connection e connessione con le credenziali attuali

```
Set objConn = CreateObject("ADODB.Connection")
objConn.Provider = "ADSDSOObject"
objConn.Open ""
```

Controlla l'esito dell'autenticazione

```
If objConn.State = adStateOpen Then
  WScript.Echo "Connection OK!" & vbCrlf
Else
  WScript.Echo "Connection NOT OK!"
```

```
   WScript.Quit(1)
End If
```

Creazione dell'oggetto Command

```
Set objCommand = CreateObject("ADODB.Command")
Set objCommand.ActiveConnection = objConn
strQuery = "SELECT Name FROM '" & strDomain & "' WHERE
ObjectCategory ='organizationalUnit'"
objCommand.CommandText = strQuery
objCommand.Properties("Sort on") = "Name"
objCommand.Properties("Size Limit") = ItemReturned
objCommand.Properties("SearchScope") = ADS_SCOPE_SUBTREE
```

Elenca tutti i container e le Organizational Unit

```
Set objRS = objCommand.Execute
While Not objRS.EOF
      Wscript.Echo objRS.Fields.Item("Name").Value
      objRS.MoveNext Wend
```

Chiusura della connessione e distruzione degli oggetti

```
objConn.Close
Set objConn = Nothing
Set objRS = Nothing
Set strDomain = Nothing
Set objRootDSE = Nothing
Set objCommand = Nothing
```

Listato 21.- Creare una nuova OU

```
Option Explicit
```

```
Dim objRootDSE
Dim objDomainDN
Dim objOUParentDN
Dim objOU
Dim strOUName
Dim strOUParentDN
Dim strOUDescription
Binding all'oggetto RootDSE
Set objRootDSE = GetObject("LDAP://rootDSE")
```

Recupero del nome di dominio che poi sarà il container per la nuova OU

```
strOUParentDN = "LDAP://" & objRootDSE.Get("defaultNamingContext")
```

Imposta il nome, il contenitore che conterrà la OU ed una breve descrizione

```
strOUName = "OU_Test"
strOUDescription = "Questa è una OU creata attraverso uno script"
Binding al container della OU
Set objOUParentDN = GetObject(strOUParentDN)
  Set objOU = objOUParentDN.Create("organizationalUnit",
 "OU=" & strOUName)
objOU.Put "description", strOUDescription
objOU.SetInfo
```

' Distruggi gli oggetti

```
Set objRootDse = Nothing
Set objDomainDN = Nothing
Set objOUParentDN = Nothing
Set objOU = Nothing
```

Listato 22.– Eliminare una OU

```
Option Explicit
Dim objRootDSE
Dim strDomainDN
Dim objOU
Dim strOUName
Binding all'oggetto RootDSE
Set objRootDSE = GetObject("LDAP://rootDSE")
```

Recupero del nome di dominio

```
strDomainDN = objRootDSE.Get("defaultNamingContext")
```

Richiedi la OU da eliminare

```
strOUName = InputBox(   "Inserire/completare la stringa per il binding
alla OU da eliminare." &  vbCrlf & "Ad
esempio: ou=OU1,dc=MyDomain,dc=it","Inserisci il
    nome della OU",strDomainDN)
 Binding alla OU specificata ed eliminazione
Set objOU = GetObject("LDAP://" & strOUName)
objOU.DeleteObject(0)
```

Distruggi oggetto

```
Set objRootDSE = Nothing
Set objOU = Nothing
```

Listato 23.– Verificare l'appartenenza del computer ad una OU

In questo script viene utilizzato un oggetto che utilizza l'interfaccia IADsADSystemInfo per recuperare diverse informazioni sul computer

corrente. In questo listato sfruttiamo questo oggetto per verificare se il computer corrente appartiene o meno ad una OU.
In testa allo script, però, è stato inserito anche un elenco delle proprietà che possiamo utilizzare attraverso l'oggetto ADSystemInfo.

```
Option Explicit
Dim strPCDN
Dim strOU
Dim objSysInfo

' Elenco attributi oggetto ADSystemInfo
' (http://www.microsoft.com/technet/scriptcenter/guide/sas_srv_gw-
qy.mspx?mfr=True)
'
' Attributo                    Descrizione
' --------------------------------------------------------------------------
--
' UserName                     Distinguished name dell'utente che
                               ha effettuato il logon.
' ComputerName                 Distinguished name del computer.
' SiteName                     Nome del sito all'interno del quale si
                               trova il PC.
' DomainShortName              Nome corto del dominio.
' DomainDNSName                DNS name del dominio.
' ForestDNSName                DNS name della foresta.
' PDCRoleOwner                 Distinguished name PDC emulator.
' SchemaRoleOwner              Distinguished name Schema master.
' IsNativeMode                 Valore booleano che indica se il do
                               minio è o meno in Native Mode.

strOU = InputBox
("Inserisci il nome di una Organizational Unit presente nel tuo dominio:")
If strOU = "" Then WScript.Quit(1)
```

```
Set objSysInfo = CreateObject("ADSystemInfo")
strPCDN = objSysInfo.ComputerName
```

Verifica se all'interno del DN del PC corrente c'è il riferimento alla OU indicata

```
If InStr(UCase(strPCDN), "OU=" & strOU) > 0 Then
 Wscript.Echo "Il Computer corrente è all'interno della OU " & strOU
Else
   Wscript.Echo "Il Computer corrente non è all'interno della OU " & strOU

End If
```

Distruggi l'oggetto

```
Set objSysInfo = Nothing
```

Listato 24.– Spostare un oggetto da una OU ad un'altra

```
Option Explicit
Dim strDestinationDN
Dim strSourceDN
Dim strSourceRDN
Dim objContainer
```

Questo esempio sposta l'utente Utente1 dalla OU1 alla OU2 all'interno del dominio MyDomain.it

Binding al container OU1

```
strDestinationDN = "LDAP://ou=OU2,dc=MyDomain,dc=it"
Set objContainer = GetObject(strDestinationDN)
```

Impostazione della stringa DN e RDN della destinazione (OU2)

```
strSourceDN = "LDAP://cn=Utente1,ou=OU1,dc=MyDomain,dc=it"
strSourceRDN = "cn=Utente1"
```

Spostamento dell'oggetto

```
objContainer.MoveHere strSourceDN, strSourceRDN
```

Distruzione degli oggetti

```
Set objContainer = Nothing
```

Listato 25.– Elencare i siti

```
Option Explicit
Dim objRootDSE
Dim strConfigNameContext
Dim strSitesContainer
Dim objSite
Dim objSitesContainer
Binding all'oggetto RootDSE

Set objRootDSE = GetObject("LDAP://RootDSE")

strConfigNameContext = objRootDSE.Get("configurationNamingContext")
Binding al Sites Container
strSitesContainer = "LDAP://cn=Sites," & strConfigNameContext
Set objSitesContainer = GetObject(strSitesContainer)
objSitesContainer.Filter = Array("site")
Wscript.Echo "--------------------"
Wscript.Echo "Elenco siti rilevati"
Wscript.Echo "--------------------"
```

L'output dei nomi rilevati è nella forma
' CN=<Site1>
' CN=<Site2>
' CN=...
' Per estrarre i nomi senza il prefisso CN= basta utilizzare Mid(objSite.Name,4)
Per ogni oggetto trovato...

```
For Each objSite In objSitesContainer
    WScript.Echo "Sito: " & Mid(objSite.Name,4)
Next
```

Distruggi gli oggetti

```
Set objRootDSE = Nothing
Set objSite = Nothing
Set objSitesContainer = Nothing
```

Listato 26.– Elencare le subnet

```
Option Explicit
Dim objRootDSE
Dim strConfigNameContext
Dim strSubnetsContainer
Dim objSubnet
Dim objSubnetsContainer
```

Binding all'oggetto RootDSE

```
Set objRootDSE =
GetObject("LDAP://RootDSE")
strConfigNameContext = objRootDSE.Get("configurationNamingContext")
```

Binding al Subnets Container

```
strSubnetsContainer =
"LDAP://cn=Subnets,cn=Sites," & strConfigNameContext
Set objSubnetsContainer = GetObject(strSubnetsContainer)
objSubnetsContainer.Filter = Array("subnet")
Wscript.Echo "----------------------"
Wscript.Echo "Elenco subnet rilevate"
Wscript.Echo "----------------------"
```

L'output dei nomi rilevati è nella forma
CN=<Subnet1>
CN=<Subnet2>
CN=...
Per estrarre i nomi senza il prefisso CN= basta utilizzare Mid(obj Subnet.Name,4)

Per ogni oggetto trovato...

```
For Each objSubnet In objSubnetsContainer
   WScript.Echo "Subnet: " & Mid(objSubnet.Name,4)
Next
```

Distruggi gli oggetti

```
Set objRootDSE = Nothing
Set objSubnet = Nothing
Set objSubnetsContainer = Nothing
```

Listato 27.– Elencare gli attributi di una classe

```
Option Explicit
```

```
Dim objClass
Dim strPropName
Dim strClass
Dim Cont
On Error Resume Next
```

Se l'utente passa da linea di comando il nome della classe

```
If Wscript.Arguments.count = 0 Then
   strUser = InputBox("Inserisci il nome della classe:")
   If strUser = "" Then WScript.Quit(1)
Else
```

altrimenti richiedilo esplicitamente

```
   strUser = Wscript.Arguments(0)
End If
strClass = Wscript.Arguments(0)
Wscript.Echo "ELENCO ATTRIBUTI CLASSE " & strClass & vbCrlf
Binding all'oggetto classe dello Schema
Set objClass = GetObject("LDAP://schema/" & strClass)
```

Visualizza tutti gli attributi obbligatori della classe

```
Cont = 0
Wscript.Echo "- OBBLIGATORI - (" &
Ubound(objClass.MandatoryProperties) + 1 & ")" & vbCrlf
For Each strPropName In objClass.MandatoryProperties
     Cont = Cont + 1
     WScript.Echo Cont & ") " & strPropName
Next
```

Visualizza tutti gli attributi opzionali della classe

```
WScript.Echo
Cont = 0

Wscript.Echo "- OPZIONALI - (" & Ubound(objClass.OptionalProperties)
+  1 & ")"& vbCrlf
For Each strPropName In objClass.OptionalProperties
    Cont = Cont + 1
    WScript.Echo Cont & ") " & strPropName
Next
```

Distruggi l'oggetto

```
Set objClass = Nothing
```

Listato 28.– Elencare gli attributi replicati

```
Option Explicit
Dim objRootDSE
Dim strSchema
Dim objCommand
Dim objConnection
Dim strQuery

Dim objRS
Dim strBase
Dim strFilter
Const ADS_SCOPE_BASE = 0
Const ADS_SCOPE_ONELEVEL = 1
Const ADS_SCOPE_SUBTREE = 2
Const PageSize = 100
Const Timeout = 30
Const CacheRes = False
```

Determina il nome dello Schema Container.

```
Set objRootDSE = GetObject("LDAP://RootDSE")
strSchema = objRootDSE.Get("schemaNamingContext")
```

Crea gli oggetti Connection e Command per il recupero delle info.

```
Set objConnection = CreateObject("ADODB.Connection")
objConnection.Provider = "ADsDSOObject"
objConnection.Open "Active Directory Provider"

Set objCommand = CreateObject("ADODB.Command")
objCommand.ActiveConnection = objConnection
```

Crea la stringa che permetterà l'avvio della ricerca con ADO.
NB: Il TRUE posto alla fine della stringa strFilter è tutto maiuscolo.
' in caso contrario la query fallisce.

```
strBase = "<LDAP://" & strSchema & ">"

strFilter =
"(&(objectCategory=attributeSchema)
(isMemberOfPartialAttributeSet=TR
UE))"
```

La prossima stringa mostra invece gli attributi non replicati strFilter = "(&(objectClass=attributeSchema)(isMemberOfPartialAttribute-Set=FALSE))"

```
strQuery = strBase & ";" & strFilter & ";cn"

objCommand.CommandText = strQuery
objCommand.Properties("Sort on") = "cn"
```

```
objCommand.Properties("Page Size") = PageSize
objCommand.Properties("SearchScope") = ADS_SCOPE_SUBTREE
objCommand.Properties("Timeout") = Timeout
objCommand.Properties("Cache Results") = CacheRes

Set objRS = objCommand.Execute
```

Mostra tutti gli attributi trovati

```
Wscript.Echo
Wscript.Echo "Nome dello Schema Container: " & strSchema
Wscript.Echo "ELENCO ATTRIBUTI TROVATI:"
WScript.Echo
While Not objRS.EOF

    Wscript.Echo objRS.Fields("cn")
    objRS.MoveNext

Wend
```

Chiudi la connessione e distruggi gli oggetti

```
objConnection.Close
Set objRootDSE = Nothing
Set objCommand = Nothing
Set objConnection = Nothing
Set objRS = Nothing
```

i libri di

LAVORARE CON ACTIVE DIRECTORY

Autore: Francesco Lippo

EDITORE
Edizioni Master S.p.A.
Sede di Milano: Via Ariberto, 24 - 20123 Milano
Sede di Rende: C.da Lecco, zona ind. - 87036 Rende (CS)

Realizzazione grafica:
Cromatika Srl
C.da Lecco, zona ind. - 87036 Rende (CS)

Art Director: Paolo Cristiano
Responsabile di progetto: Salvatore Vuono
Coordinatore tecnico: Giancarlo Sicilia
Illustrazioni: Tonino Intieri
Impaginazione elettronica: Salvatore Spina

Servizio Clienti
☎ **Tel. 02 831212 - Fax 02 83121206**
@ **e-mail: customercare@edmaster.it**

Stampa: Grafica Editoriale Printing - Bologna

Finito di stampare nel mese di Luglio 2006